

(12)特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関  
国際事務局(43) 国際公開日  
2004 年 10 月 21 日 (21.10.2004)

PCT

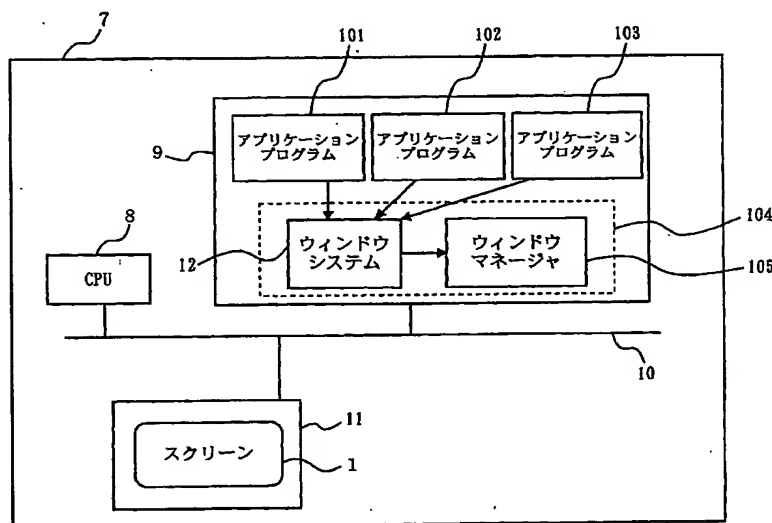
(10) 国際公開番号  
WO 2004/090712 A1

- (51) 国際特許分類: G06F 3/14, G09G 5/14 (72) 発明者; および  
(21) 国際出願番号: PCT/JP2004/004414 (75) 発明者/出願人 (米国についてのみ): 瀬本 雄二  
(22) 国際出願日: 2004 年 3 月 29 日 (29.03.2004) (74) 代理人: 小笠原 史朗 (OGASAWARA, Shiro); 〒  
(25) 国際出願の言語: 日本語 5640053 大阪府吹田市江の木町 3 番 1 1 号 第 3 ロン  
(26) 国際公開の言語: 日本語 チェビル Osaka (JP).  
(30) 優先権データ: 特願2003-106393 2003 年 4 月 10 日 (10.04.2003) JP  
(71) 出願人 (米国を除く全ての指定国について): 松下電  
器産業株式会社 (MATSUSHITA ELECTRIC INDUS-  
TRIAL CO., LTD.) [JP/JP]; 〒5718501 大阪府門真市大  
字門真 1 0 0 6 Osaka (JP).  
(81) 指定国 (表示のない限り、全ての種類の国内保護が  
可能): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR,  
BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM,  
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU,  
ID, IL, IN, IS, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT,  
LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI,  
NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG,  
SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ,  
VC, VN, YU, ZA, ZM, ZW.

/続葉有/

(54) Title: WINDOW STACK CONTROL METHOD, WINDOW MANAGEMENT PROGRAM, AND WINDOW MANAGE-  
MENT APPARATUS

(54) 発明の名称: ウィンドウスタック制御方法、ウィンドウ管理プログラム、ウィンドウ管理装置



101...APPLICATION PROGRAM  
102...APPLICATION PROGRAM  
103...APPLICATION PROGRAM  
12...WINDOW SYSTEM  
105...WINDOW MANAGER  
1...SCREEN

(57) Abstract: When controlling a window stack that manages a stack of windows during displaying, on a display device (11), a plurality of windows from a computer (7), application programs (101,102,103) specify groups for the windows, and a window management program (104) manages that the stack order of the windows is series in each group. This prevents a window display effected by an application displaying a window on the top from being accidentally influenced by another application.

(57) 要約: コンピュータ (7) から表示装置 (11) に複数のウィンドウを表示する際のウィンドウの重ね合わせを管理するウィンドウスタックを制御するに際し、アプリケーションプログラム (101, 102, 103) がウィンドウにグループを指定し、ウィンドウ管理プログラム (104) がウィンドウのスタック順をグループ毎に一連とする。これにより、最前面でウィンドウを表示中のアプリケーションのウィンドウ表示に、他のアプリケーションが不用意に影響を及ぼす事が無くなる。



(84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

添付公開書類:

- 国際調査報告書
- 補正書

2文字コード及び他の略語については、定期発行される各PCTガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

## 明 細 書

ウィンドウスタック制御方法、ウィンドウ管理プログラム、ウィンドウ  
管理装置

### 技 術 分 野

本発明は、コンピュータから表示装置に複数のウィンドウを表示する際のウィンドウの重ね合わせを管理するウィンドウスタック制御方法に関する。

### 背 景 技 術

近年、表示装置に複数のウィンドウを表示するコンピュータとして、ワークステーションや家庭用コンピュータが広く普及している。これらの場合、ユーザは、表示装置として比較的大きな画面に複数のウィンドウを表示して、マウスとキーボードを使ってウィンドウの重なるの上下関係であるスタック順の操作を行うのが一般的である。

例えば、Unix (R) ワークステーションで広く普及している X 1 1 ウィンドウシステムの場合、ウィンドウの重なるの上下関係であるスタック順は、ウィンドウの生成時にスタック最上位に位置付けられ、マップされた後は、マウスやアプリケーションの操作によって、スタック位置を上下方向に移動させる。

具体例として、コンピュータ上で動作する 3 種類のアプリケーションが合計 5 枚のウィンドウを表示装置のスクリーンに表示する際の表示例となる模式図を図 2 9 に示す。図 2 9 において、1 は C R T モニタや液晶モニタで実現さ

れる表示装置のスクリーン、2及び3は第1のアプリケーションが表示するウィンドウ、4及び5は第2のアプリケーションが表示するウィンドウ、6は第3のアプリケーションが表示するウィンドウであり、第1のアプリケーションが表示するウィンドウ2及びウィンドウ3を第1のグループ、第2のアプリケーションが表示するウィンドウ4及びウィンドウ5を第2のグループ、第3のアプリケーションが表示するウィンドウ6を第3のグループとする。

図29を見ればわかるように、ウィンドウ3はウィンドウ2の上に重なり、ウィンドウ5はウィンドウ4の上に重なっている。この時のウィンドウスタック順の例となる模式図を図30に示す。図30を見ればわかるように、ウィンドウ3はウィンドウ2よりスタック上位にあるので、両者が表示上で重なる場合は、ウィンドウ3がウィンドウ2の上に重なる事となり、同様に、ウィンドウ5はウィンドウ4よりスタック上位にあるので、両者が表示上で重なる場合は、ウィンドウ5がウィンドウ4の上に重なって表示される。

そして、このような表示状態、スタック順状態で、第1のアプリケーションが何らかの都合で、例えば、ウィンドウ2をウィンドウ3の前面に表示させたくなった場合、ウィンドウ2をスタック最上位に持ってくる事で対応していた。この時のスタック順の模式図を図31に示す。図31を見ればわかるように、ウィンドウ2はウィンドウ3よりスタック上位にあるので、両者が表示上で重なる場合は、ウィンドウ2がウィンドウ3の上に重なって表示されるこ

ととなる。また、この操作により、ウィンドウ 2 が、第 2 のグループ、第 3 のグループのウィンドウを跨って、スタック順で最下位から最上位へ移動した事がわかる。

### 発明の開示

しかしながら、上記従来のウィンドウスタック制御方法においては、アプリケーションが或るウィンドウに対して、ウィンドウスタック移動、ウィンドウ生成及び表示を行った場合、グループを跨ってスタック制御をするため、意図せず、該ウィンドウが他のアプリケーションの表示するウィンドウの上に重なることがあるという問題があった。

上記課題に鑑み、本発明は、ウィンドウスタックの制御方法において、最前面でウィンドウを表示中のアプリケーションのウィンドウ表示に、他のアプリケーションが不用意に影響を及ぼす事の無いウィンドウスタック制御方法を提供することを目的とする。

この課題を解決するために本発明は以下の構成を採用した。なお、括弧内の参照符号および図番号は、本発明の理解を助けるために図面との対応関係を示したものであって、本発明の範囲を何ら限定するものではない。

本発明の第 1 の発明は、1 つ以上のアプリケーションプログラム（101，102，103）に基づいて表示装置（11）に表示される複数のウィンドウの重ね合わせを管理するためのウィンドウスタック制御方法であって、アプリケーションプログラムからウィンドウのグループの指定を受けるステップと、アプリケーションプログラムからウ

インドウの表示要求を受けるステップと、表示要求を受けて当該ウィンドウの表示を行う際に、当該ウィンドウのスタック順をグループ毎に一連となるように纏めるステップ（S 3 0 8，図 4，S 2 5 1 0，図 2 6）とを備えたウィンドウスタック制御方法である。

これにより、スタック順がグループ単位で纏められるため、最上位のグループに属する各ウィンドウは、他のグループのウィンドウの下に重ねられて表示される事が無くなり、最前面でウィンドウを表示中のアプリケーションのウィンドウ表示に、他のアプリケーションが不用意に影響を及ぼす事を無くす事ができる。特に、画面が小さく、結果として、ウィンドウの重なり易い携帯電話や P D A 等で、その効果は顕著である。

また、本発明の第 2 の発明は、第 1 の発明において、前記纏めるステップは、グループ毎に纏めていない第 1 のウィンドウの表示要求をアプリケーションプログラムから受け取った際に、グループ毎に纏めていないウィンドウの中で前記第 1 のウィンドウと同じグループに属する第 1 のウィンドウ群のスタック順が、該第 1 のウィンドウ群の中でのスタック順関係は保持したまま、既にグループ毎に纏められたウィンドウの中で前記第 1 のウィンドウと同じグループに属する第 2 のウィンドウ群のスタック順と一連となるようにする事を特徴とするウィンドウスタック制御方法である（図 2 6）。

これにより、表示要求によって同一グループ内でみたスタック順が変更されることがないので、アプリケーション

の同一グループ内のスタック順管理の負担が軽減される。

また、本発明の第3の発明は、第1又は第2の発明において、前記ウィンドウスタック制御方法は、グループ毎に代表ウィンドウを1枚生成するステップをさらに備え、前記纏めるステップは、ウィンドウのスタック順をグループ毎に一連とする際に、当該ウィンドウを前記代表ウィンドウの子ウィンドウとする事を特徴とするウィンドウスタック制御方法である（図22）。

これにより、グループ単位のスタック移動やグループ単位のウィンドウの表示／非表示の切り替えを代表ウィンドウに対する制御で実施できる分、グループ内のウィンドウが多いほど、処理負荷を軽減することができるようになる。

また、本発明の第4の発明は、第1から第3のいずれかの発明において、アプリケーションプログラムからグループ内でのウィンドウスタックの最上位移動要求および最下位移動要求を受けるステップと、前記最上位移動要求または前記最下位移動要求を受けて、グループ内でのスタック変更を行うステップ（S708，S710，S711，S1208，S1210，S1211）とをさらに備えたウィンドウスタック制御方法である。

これにより、スタックの操作がグループ内で閉じるので、各アプリケーションプログラムは、他のアプリケーションプログラムのウィンドウ表示を気にする事無くスタック変更を行う事ができる。

また、本発明の第5の発明は、第4の発明において、前

記スタック変更を行うステップは、前記第 1 のウィンドウに対するグループ内でのウィンドウスタックの最上位移動要求および最下位移動要求をアプリケーションプログラムから受け取った際に、グループ毎に纏めていないウィンドウの中で前記第 1 のウィンドウと同じグループに属する第 1 のウィンドウ群のスタック順が、前記第 1 のウィンドウ群の中でのスタック順関係は保持したまま、既にグループ毎に纏められたウィンドウの中で前記第 1 のウィンドウと同じグループに属する第 2 のウィンドウ群のスタック順と一連となるようにスタック変更を行うことを特徴とする（S 7 0 8，図 9，S 1 2 0 8，図 1 4）ウィンドウスタック制御方法である。

これにより、スタック変更対象となるウィンドウと同じグループに属する通常ウィンドウ全てが纏められるため、アプリケーションプログラムは、自身が設定したグループのウィンドウのスタック順を把握し易くなる。

また、本発明の第 6 の発明は、第 1 から第 3 のいずれかの発明において、アプリケーションプログラムからグループ単位でのウィンドウスタックの最上位移動要求および最下位移動要求を受けるステップと、前記最上位移動要求または前記最下位移動要求を受けて、グループ単位でのスタック変更を行うステップ（S 1 7 0 5，S 1 7 0 6，S 2 0 0 4）とをさらに備えたウィンドウスタック制御方法である。

これにより、ウィンドウの操作をグループ指定で実施できるので、例えば、グループをアプリケーションプログラ



ム単位で振り分けた場合だと、そのアプリケーションに属するウィンドウのメンバーを知らなくても、スタック最前面にそのアプリケーションに属する全通常ウィンドウを移動させる事が可能となる。

また、本発明の第7の発明は、第6の発明において、前記スタック変更を行うステップは、第1のグループに対するグループ単位でのウィンドウスタックの最上位移動要求および最下位移動要求をアプリケーションプログラムから受け取った際に、グループ毎に纏めていないウィンドウの中で前記第1のグループに属する第1のウィンドウ群のスタック順が、前記第1のウィンドウ群の中でのスタック順関係は保持したまま、既にグループ毎に纏められたウィンドウの中で前記第1のグループに属する第2のウィンドウ群のスタック順と一連となるようにスタック変更を行うことを特徴とする（S1705，図18，S2004，図21）ウィンドウスタック制御方法である。

これにより、スタック変更対象となるグループに属する通常ウィンドウ全てを、その中でのスタック順は保持したままで纏めた上でスタック変更を実施するので、アプリケーションプログラムは、自身が設定したグループのウィンドウのスタック順を把握し易くなる。

また、本発明の第8の発明は、第1の発明において、アプリケーションプログラムによって優先ウィンドウとして指定されたウィンドウをいずれのグループにも所属させず、かつ当該優先ウィンドウを、表示装置に表示された、いずれかのグループに所属する全てのウィンドウより常にス

タック上位に位置させることを特徴とするウィンドウスタック制御方法である。

これにより、グループ単位のスタック順に関係無くウィンドウを表示できるので、最優先で表示するウィンドウのサービスを提供することができる。

また、本発明の第 9 の発明は、第 1 の発明において、最上位グループに属するウィンドウよりスタック下位のウィンドウを常に非表示状態とすることを特徴とするウィンドウスタック制御方法である。

これにより、最上位グループよりスタック下位のグループが何を表示しようとしても表示装置には表示されないのので、最上位グループのウィンドウで覆われない部分に中途半端に映り込む心配が無い。

また、本発明の第 10 の発明は、第 1 の発明において、最上位グループに属するウィンドウの中のスタック最下位のウィンドウの直下に、特定のウィンドウを位置させることを特徴とする（図 23，図 24）ウィンドウスタック制御方法である。

これにより、例えば、前記特定のウィンドウを表示装置のスクリーン一杯に表示しておけば、前記特定のウィンドウのスタック下位のグループが何を表示しようとしても、表示装置に表示されないのので、最上位グループのウィンドウで覆われない部分に中途半端に映り込む心配が無い。

また、本発明の第 11 の発明は、第 4 又は第 6 の発明において、前記ウィンドウスタック制御方法は、Xウィンドウシステムとウィンドウマネージャによって複数のウィン

ドウの重ね合わせを管理するものであって、最上位グループの直上に特定のウィンドウを位置させることを特徴とするウィンドウスタック制御方法である。

これにより、最上位グループを別のグループに入れ替える際、Xウィンドウシステムの場合、X R e s t a c k W i n d o w s ( ) 関数を使用する事となるが、ウィンドウ群のスタック移動先を示すために使用できる。もし、最上位グループのウィンドウよりスタック上位にウィンドウが存在しなかった場合、存在する全てのウィンドウの並び順を決め、前記 X R e s t a c k W i n d o w s ( ) の引数に代入せねばならないが、前記特定のウィンドウがあれば、前記特定のウィンドウと前記ウィンドウ群を引数に代入するだけで済む。

また、本発明の第 1 2 の発明は、第 1 から第 3 のいずれかの発明において、前記ウィンドウスタック制御方法は、Xウィンドウシステムとウィンドウマネージャによって複数のウィンドウの重ね合わせを管理するものであって、ウィンドウマネージャが、ウィンドウ破棄通知受信時に、ウィンドウシステム間とでスタック認識の整合性を確認し、異なっている場合は、ウィンドウシステムのスタック認識をウィンドウマネージャのスタック認識に合わせ込む処理を行うことを特徴とするウィンドウスタック制御方法である。

これにより、ウィンドウ破棄通知受信時に整合性確保を図るので、ウィンドウシステムがスタック変更処理に失敗しても、処理が破綻する事がなくなる。

また、本発明の第 13 の発明は、第 1 から第 3 のいずれかの発明において、前記ウィンドウスタック制御方法は、Xウィンドウシステムとウィンドウマネージャによって複数のウィンドウの重ね合わせを管理するものであって、ウィンドウマネージャが、ウィンドウシステムにスタック変更を要求する際にフラグを立てる一方で、ウィンドウ破棄通知受信時に、前記フラグが立っている時のみ、ウィンドウシステム間とでスタック認識の整合性を確認し、異なっている場合は、ウィンドウシステムのスタック認識をウィンドウマネージャのスタック認識に合わせ込む処理を行い、前記フラグを下げることを特徴とするウィンドウスタック制御方法である。

これにより、ウィンドウマネージャがスタック変更を実施した時のみ、スタック認識の整合性のための処理を行うので、処理の実行頻度を軽減できる。

また、本発明の第 14 の発明は、第 1 の発明において、前記纏めるステップは、グループ毎に纏めていない第 1 のウィンドウの表示要求をアプリケーションプログラムから受け取った際に、既にグループ毎に纏められたウィンドウの中で前記第 1 のウィンドウと同じグループに属する第 2 のウィンドウ群のスタック順と一連となるようにすることを特徴とするウィンドウスタック制御方法である（図 4）。

また、本発明の第 15 の発明は、1 つ以上のアプリケーションプログラム（101，102，103）に基づいて表示装置（11）に表示される複数のウィンドウの重ね合

わせを管理するためのウィンドウ管理プログラム（１０４）であって、コンピュータ（７）に、アプリケーションプログラムからウィンドウのグループの指定を受けるステップと、アプリケーションプログラムからウィンドウの表示要求を受けるステップと、表示要求を受けた際に該ウィンドウのスタック順をグループ毎に一連となるように纏めるステップ（Ｓ３０８，図４）とを実行させるためのウィンドウ管理プログラムである。

また、本発明の第１６の発明は、表示装置（１１）に複数のウィンドウを表示する際のウィンドウの重ね合わせを管理するウィンドウ管理装置であって、１つ以上のウィンドウを前記表示装置に表示する１つ以上のアプリケーションプログラム（１０１，１０２，１０３）と、前記１つ以上のアプリケーションプログラムが表示するウィンドウの重ね合わせを管理するウィンドウ管理プログラム（１０４）と、前記アプリケーションプログラムおよび前記ウィンドウ管理プログラムを実行する処理部（８）とを備え、前記アプリケーションプログラムは、前記ウィンドウ管理プログラムに対してウィンドウのグループを指定するものであり、前記ウィンドウ管理プログラムは、前記アプリケーションプログラムからウィンドウの表示要求を受けて当該ウィンドウの表示を行う際に、当該ウィンドウのスタック順をグループ毎に一連とする制御を行う（Ｓ３０８，図４）ものであることを特徴とするウィンドウ管理装置である。

## 図面の簡単な説明

図 1 は、本発明の実施の形態 1 のウィンドウスタック制御方法を実現するシステムの構成例を示す図である。

図 2 は、本発明実施の形態 1 のウィンドウ情報の一例を示す図である。

図 3 は、本発明実施の形態 1 のマップ要求イベント (Map Request) を受け取った際のウィンドウマネージャの動作例となるフロー図である。

図 4 は、本発明の実施の形態 1 の図 3 のステップ S 3 0 8 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 5 は、本発明の実施の形態 1 の図 3 のステップ S 3 1 0 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 6 は、本発明の実施の形態 1 の図 3 のステップ S 3 1 1 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 7 は、本発明の実施の形態 1 のスタック最上位移動要求イベントを受け取った際のウィンドウマネージャの動作例となるフロー図である。

図 8 は、本発明の実施の形態 1 の図 7 のステップ S 7 0 5 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 9 は、本発明の実施の形態 1 の図 7 のステップ S 7 0 8 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 1 0 は、本発明の実施の形態 1 の図 7 のステップ S 7 1 0 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 1 1 は、本発明の実施の形態 1 の図 7 のステップ S 7 1 1 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 1 2 は、本発明の実施の形態 1 のスタック最下位移動要求イベントを受け取った際のウィンドウマネージャの動作例となるフロー図である。

図 1 3 は、本発明の実施の形態 1 の図 1 2 のステップ S 1 2 0 5 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 1 4 は、本発明の実施の形態 1 の図 1 2 のステップ S 1 2 0 8 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 1 5 は、本発明の実施の形態 1 の図 1 2 のステップ S 1 2 1 0 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 1 6 は、本発明の実施の形態 1 の図 1 2 のステップ S 1 2 1 1 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 1 7 は、本発明の実施の形態 1 のグループ単位スタック最上位移動要求メッセージを受け取った際のウィンドウマネージャの動作例となるフロー図である。

図 1 8 は、本発明の実施の形態 1 の図 1 7 のステップ S 1 7 0 5 のスタック変更要求によってスタックがどのように

に変化するかを示す具体例となる模式図である。

図 1 9 は、本発明の実施の形態 1 の図 1 7 のステップ S 1 7 0 6 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 2 0 は、本発明の実施の形態 1 のグループ単位スタック最下位移動要求メッセージを受け取った際のウィンドウマネージャの動作例となるフロー図である。

図 2 1 は、本発明の実施の形態 1 の図 2 0 のステップ S 2 0 0 4 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 2 2 は、本発明の実施の形態 1 の代表ウィンドウ使用下でのスタックの様子を示す例となる模式図である。

図 2 3 は、本発明の実施の形態 1 の最上位グループに属する全ウィンドウの中のスタック最下位のウィンドウの直下に、ウィンドウ管理プログラムが生成及びマップする特定のウィンドウを位置させて使用した場合のスタックの様子を示す例となる模式図である。

図 2 4 は、本発明の実施の形態 1 の最上位グループに属する全ウィンドウの中のスタック最下位のウィンドウの直下に、ウィンドウ管理プログラムが生成及びマップする特定のウィンドウを位置させて使用した場合のスクリーンに表示されている例となる模式図である。

図 2 5 は、本発明の実施の形態 2 のマップ要求イベント ( M a p R e q u e s t ) を受け取った際のウィンドウマネージャの動作例となるフロー図である。

図 2 6 は、本発明の実施の形態 2 の図 2 5 のステップ S



2 5 1 0 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 2 7 は、本発明の実施の形態 2 の図 2 5 のステップ S 2 5 1 2 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 2 8 は、本発明の実施の形態 2 の図 2 5 のステップ S 2 5 1 3 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図である。

図 2 9 は、従来例におけるコンピュータ上で動作する 3 種類のアプリケーションが合計 5 枚のウィンドウを表示装置であるスクリーンに表示する際の表示例となる模式図である。

図 3 0 は、従来例におけるウィンドウスタック順の例となる模式図である。

図 3 1 は、従来例におけるウィンドウスタック順の例となる模式図である。

### 発明を実施するための最良の形態

以下、本発明の実施の形態について、図面を用いて説明する。

#### (実施の形態 1)

本実施の形態では、ウィンドウ管理プログラムが、X ウィンドウシステムとウィンドウマネージャから成り、ウィンドウマネージャがルートウィンドウの子ウィンドウであるトップレベルウィンドウのスタック順を制御する場合を例にとって説明する。

図 1 に本発明の実施の形態 1 のウィンドウスタック制御方法を実現するシステムの構成例を示す。

図 1 において、7 はコンピュータである。8 は CPU (Central Processing Unit) である。9 は各種プログラムや処理データ等を格納するメモリである。10 は各構成要素に所望の双方向通信を行わせるバスである。11 は CRT モニタや液晶モニタといった表示装置である。1 は表示装置 11 のスクリーンである。101, 102 及び 103 は CPU 8 で実行されるプログラムであって、1 枚以上のウィンドウをウィンドウ管理プログラム 104 を介してスクリーン 1 に表示しようとするアプリケーションプログラムである。104 は CPU 8 で実行されるプログラムであって、アプリケーションプログラム 101 ~ 103 が表示しようとするウィンドウを管理してバス 10 を介して表示を行うウィンドウ管理プログラムである。ウィンドウ管理プログラム 104 は、ウィンドウシステム 12 とウィンドウマネージャ 105 から成る。ウィンドウシステム 12 は従来の X ウィンドウシステムであり、105 はウィンドウマネージャである。

以下、図面を用いて本実施の形態の動作を説明する。

アプリケーションプログラム 101、102、103 は、ウィンドウ管理プログラム 104 を通じて、スタック制御特性の異なる 2 種類のトップレベルウィンドウを表示することができる。その 2 種類を通常ウィンドウ、優先ウィンドウと呼ぶ事とする。両者の違いは、表示された優先ウィンドウは表示された全ての通常ウィンドウより常にスタ

ック上位に表示されることと、通常ウィンドウはグループ単位でスタック順を一連とするが優先ウィンドウについてはそのような処理を行わないことである。

＜ウィンドウ生成要求，ウィンドウ種類及びグループ設定要求，マップ要求＞

アプリケーションプログラム 101、102、103は、ウィンドウを表示する場合、ウィンドウシステム 12に対して、ウィンドウ生成要求と、ウィンドウ種類及びグループ設定要求と、マップ要求とを発行する。この内、ウィンドウ種類及びグループ設定要求については、Xウィンドウシステムの場合、プロパティ機構を用いる事で、ウィンドウ毎に値を設定することができる。

各ウィンドウは、ウィンドウ生成及びマップという過程を経て初めてスクリーン 1 に表示されるようになる。

ウィンドウシステム 12 は、ウィンドウ生成要求を受け取ると、ウィンドウの重なりの順番であるスタック順の最上位にそのウィンドウを位置させ、ウィンドウマネージャ 105 に対してウィンドウ生成通知イベント（Create Notify）を発行する。

ウィンドウマネージャ 105 は、内部データベースとして、ウィンドウ毎にウィンドウ情報を保持する。

図 2 にウィンドウ情報の一例を示す。

図 2 に示されるように、ウィンドウ情報は、ウィンドウ識別子、スタック位置確定フラグ、スタック位置情報、プロパティ獲得フラグ、ウィンドウ種類、グループ値から成る。ウィンドウ識別子は、そのウィンドウ情報がどのウィ

ンドウの情報かを示す。スタック位置確定フラグは、そのウィンドウが、ウィンドウマネージャ 105 にとって、スタック位置が暫定か確定済みなのかを示す。スタック位置情報は、そのウィンドウのスタック位置を示す。プロパティ獲得フラグは、そのウィンドウのウィンドウ種類とグループ値をウィンドウマネージャ 105 が獲得済みかどうかを示す。ウィンドウ種類は、前記プロパティ獲得フラグが獲得済みとなっている場合は、そのウィンドウが優先ウィンドウか通常ウィンドウのどちらであるかを示す。グループ値は、前記プロパティ獲得フラグが獲得済みとなっている場合は、そのウィンドウのグループ値を示す。

図 1 のウィンドウマネージャ 105 は、ウィンドウ生成通知イベント (CreateNotify) を受け取ることで、新規にウィンドウが生成された事を知り、そのウィンドウ生成通知イベント (CreateNotify) からウィンドウ識別子を獲得する。そして、内部データベースに新規にウィンドウ情報格納領域を設け、そのウィンドウ情報のウィンドウ識別子に格納する。さらにウィンドウマネージャ 105 は、同ウィンドウのスタック位置確定フラグを暫定に、スタック位置情報をスタック最上位に、プロパティ獲得フラグを未獲得に、各々初期化する。

次に、ウィンドウシステム 12 は、ウィンドウ種類及びグループ設定要求を受け取り、該ウィンドウのプロパティとして記憶する。

次に、ウィンドウシステム 12 は、アプリケーションプログラム 101、102、103 からマップ要求を受け取

り、内部処理は特にせずに、ウィンドウマネージャ 105 にマップ要求イベント (Map Request) を発行する。

マップ要求イベント (Map Request) を受け取った際のウィンドウマネージャ 105 の動作例となるフロー図を図 3 に示す。

図 3 に示されるように、ステップ S 301 では、ウィンドウマネージャ 105 は、該時点で受け取ったマップ要求イベント (Map Request) からマップ対象ウィンドウ (以下、単に対象ウィンドウと称す) の識別子を獲得して、ステップ S 302 に進む。

ステップ S 302 では、対象ウィンドウの識別子で内部データベースを検索し、対応するウィンドウ情報が存在する場合はステップ S 303 へ進み、存在しない場合はステップ S 313 へ進む。

ステップ S 303 では、内部データベースで対象ウィンドウのスタック位置確定フラグが確定済みかどうかをチェックし、確定済みの場合はステップ S 312 に進み、暫定の場合はステップ S 304 に進む。

ステップ S 304 では、対象ウィンドウのプロパティ獲得フラグをチェックし、獲得済みの場合はステップ S 306 に進み、未獲得の場合はステップ S 305 に進む。

ステップ S 305 では、対象ウィンドウのウィンドウ種類及びグループ値をウィンドウシステム 12 に問い合わせて獲得し、内部データベースに格納し、ステップ S 306 に進む。

ステップ S 3 0 6 では、対象ウィンドウのウィンドウ種類をチェックし、優先ウィンドウの場合はステップ S 3 1 2 に進み、通常ウィンドウの場合はステップ S 3 0 7 に進む。

ステップ S 3 0 7 では、内部データベースに、対象ウィンドウと同じグループ値を持つスタック位置確定フラグが確定済みの通常ウィンドウがあるかどうかをチェックし、有る場合はステップ S 3 0 8 に進み、無い場合はステップ S 3 0 9 に進む。

ステップ S 3 0 8 では、対象ウィンドウのスタック位置が、同じグループ内のスタック位置確定済みの通常ウィンドウの中でスタック最上位であるウィンドウの直上となるように、ウィンドウシステム 1 2 にスタック変更要求を発行する。

ステップ S 3 0 8 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 4 に示す。

図 4 に示されるように、通常ウィンドウである対象ウィンドウが、対象ウィンドウと同じグループ値を持つスタック位置確定フラグが確定済みのウィンドウの内のスタック最上位のウィンドウの直上に移されているのが解る。

ステップ S 3 0 9 では、内部データベースから、通常ウィンドウ全体を対象にスタック位置確定フラグが確定済みのウィンドウを検索し、存在する場合はステップ S 3 1 0 に進み、存在しない場合はステップ S 3 1 1 に進む。

ステップ S 3 1 0 では、内部データベースにあるスタック

ク位置確定フラグが確定済みの通常ウィンドウのうち、スタック位置が最上位のウィンドウの直上に対象ウィンドウが位置するよう、ウィンドウシステム 1 2 に対してスタック変更要求を発行し、ステップ S 3 1 2 に進む。

ステップ S 3 1 0 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 5 に示す。

図 5 に示されるように、通常ウィンドウである対象ウィンドウが、スタック位置確定フラグが確定済みの通常ウィンドウ全体の内のスタック最上位のウィンドウの直上に移されているのが解る。

ステップ S 3 1 1 では、スタック位置の最下位に対象ウィンドウが位置するよう、ウィンドウシステム 1 2 にスタック変更要求を発行し、ステップ S 3 1 2 に進む。

ステップ S 3 1 1 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 6 に示す。

図 6 に示されるように、通常ウィンドウである対象ウィンドウが、スタック最下位に移されているのが解る。

ステップ S 3 1 2 では、ウィンドウシステム 1 2 に対して対象ウィンドウのマップ要求を発行し、内部データベースの対象ウィンドウのウィンドウ情報の内、スタック位置確定フラグが暫定を示している場合は確定済みに更新し、また、内部データベース全体に対して必要に応じてスタック位置情報を更新した上で、ステップ S 3 1 3 に進む。

ステップ S 3 1 3 では、処理を終了する。

ウィンドウシステム 12 は、図 3 のステップ S 308、ステップ S 310、ステップ S 311 に起因するスタック変更要求をウィンドウマネージャ 105 から受け取ると、要求に従ってウィンドウのスタック順を変更する。また、図 3 のステップ S 312 に起因するマップ要求をウィンドウマネージャ 105 から受け取ると、要求に従ってウィンドウのマップする。マップされたウィンドウは、ウィンドウの重なり具合にもよるが、他のウィンドウの下に隠れていなければ、スクリーン 1 に表示されるようになる。

図 3 の処理によって、アプリケーション 101、102、103 が生成及びウィンドウ種類及びグループ設定した通常ウィンドウは、表示までに必ず、スタック位置情報が確定済みとなり、また、スタック位置がグループ毎に纏められ、また、優先ウィンドウのスタック上位に表示される事もない。また、優先ウィンドウは、スタック位置が変更されることなく表示される。

#### <グループ内スタック最上位移動要求>

図 1 において、アプリケーションプログラム 101、102、103 は、生成及びウィンドウ種類及びグループ設定した通常ウィンドウに対して、スタック変更要求として、グループ内スタック最上位移動要求をウィンドウシステム 12 に対して発行する事ができる。Xウィンドウシステムの場合、例えば、グループ内スタック最上位移動要求に `XRaiseWindow()` 関数を割り当てれば良い。また、優先ウィンドウに対して、スタック変更要求として、優先ウィンドウスタック最上位移動要求をウィンドウシ



システム 1 2 に対して発行する事ができる。例えば、グループ内スタック最上位移動要求と同じ `X R a i s e W i n d o w ( )` 関数を割り当てれば良い。

アプリケーションプログラムが `X R a i s e W i n d o w ( )` を実施した場合、ウィンドウシステム 1 2 はスタック変更をせず、ウィンドウマネージャ 1 0 5 に対してスタック最上位移動要求イベント (`C o n f i g u r e R e q u e s t`) を発行する。

スタック最上位移動要求イベントを受け取った際のウィンドウマネージャ 1 0 5 の動作例となるフロー図を図 7 に示す。

図 7 に示されるように、ステップ S 7 0 1 では、該時点で受け取ったスタック最上位移動要求イベント (`C o n f i g u r e R e q u e s t`) から対象ウィンドウの識別子を獲得して、ステップ S 7 0 2 に進む。

ステップ S 7 0 2 では、対象ウィンドウの識別子で内部データベースを検索し、対応するデータが存在する場合はステップ S 7 0 3 へ進み、存在しない場合はステップ S 7 1 2 へ進む。

ステップ S 7 0 3 では、内部データベースにあるプロパティ獲得フラグが未獲得である全ウィンドウについて、ウィンドウシステム 1 2 に対してウィンドウ種類及びグループ値を問い合わせ、獲得できたウィンドウについては、そのウィンドウ種類及びグループ値と、プロパティ獲得済みである事を内部データベースに記憶し、ステップ S 7 0 4 に進む。

ステップ S 7 0 4 では、対象ウィンドウのウィンドウ種類をチェックし、優先ウィンドウであればステップ S 7 0 5 に進み、通常ウィンドウであればステップ S 7 0 6 に進む。

ステップ S 7 0 5 では、内部データベースからウィンドウ種類が優先ウィンドウであるものの内、スタック最上位のウィンドウを検索する。そして、対象ウィンドウが、そのスタック最上位のウィンドウの直上となるよう、ウィンドウシステム 1 2 にスタック変更要求を発行し、次に、内部データベース全体に対してスタック位置情報を更新した上で、ステップ S 7 1 2 に進む。なお、検索の結果、スタック最上位のウィンドウが対象ウィンドウ自身であった場合は、何もせずステップ S 7 1 2 に進む。

ステップ S 7 0 5 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 8 に示す。

図 8 に示されるように、対象ウィンドウのみが、優先ウィンドウの中のスタック最上位のウィンドウの直上に移されているのが解る。

図 7 において、ステップ S 7 0 6 では、対象ウィンドウと同じグループに属する通常ウィンドウを内部データベースから検索してリストアップし、ステップ S 7 0 7 に進む。

ステップ S 7 0 7 では、前記リストアップした中にスタック位置確定フラグが確定済みであるウィンドウが存在するならばステップ S 7 0 8 に進み、存在しないならステッ

プ S 7 0 9 に進む。

ステップ S 7 0 8 では、リストアップした中でスタック位置確定フラグが暫定であるウィンドウがあれば、それら全てを、それらの中でみたスタック順はそのまま、リストアップした中でスタック位置確定フラグが確定済みであるウィンドウの内の最上位のウィンドウの直上に位置させ、更に対象ウィンドウをリストアップしたウィンドウの中で最上位となるよう、ウィンドウシステム 1 2 にスタック変更要求を発行し、次に、リストアップした中でスタック位置確定フラグが暫定であるウィンドウがあれば確定済みに更新し、また、内部データベース全体に対してスタック位置情報を更新した上で、ステップ S 7 1 0 に進む。

ステップ S 7 0 8 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 9 に示す。

図 9 に示されるように、対象ウィンドウと同じグループ値を持つスタック位置確定フラグが暫定のウィンドウが、同じグループ値を持つスタック位置確定フラグが確定済みのウィンドウの直上に移され、同一グループ値を持つウィンドウ全体の中で対象ウィンドウのスタック最上位移動がなされているのが解る。

図 7 において、ステップ S 7 0 9 では、内部データベースにスタック位置確定フラグが確定済みのウィンドウが存在するならばステップ S 7 1 0 に進み、存在しないならばステップ S 7 1 1 に進む。

ステップ S 7 1 0 では、ステップ S 7 0 6 でリストアッ

プした中でスタック位置確定フラグが暫定であるウィンドウ全てを、それらの中でみたスタック順はそのまま、内部データベースの中でスタック位置確定フラグが確定済みであるスタック最上位の通常ウィンドウの直上に位置させ、更に対象ウィンドウが前記リストアップしたウィンドウの中で最上位となるよう、ウィンドウシステム 12 にスタック変更要求を発行する。そして、リストアップしたウィンドウのスタック位置確定フラグを確定済みに更新し、さらに、内部データベース全体に対してスタック位置情報を更新した上で、ステップ S 7 1 2 に進む。

ステップ S 7 1 0 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 1 0 に示す。

図 1 0 に示されるように、対象ウィンドウと同じグループ値を持つスタック位置確定フラグが暫定の通常ウィンドウが、スタック位置確定フラグが確定済みの通常ウィンドウ全体のスタック最上位ウィンドウの直上に移され、同一グループ値を持つウィンドウ全体の中で対象ウィンドウのスタック最上位移動がなされているのが解る。

図 7 において、ステップ S 7 1 1 では、ステップ S 7 0 6 でリストアップした中でスタック位置確定フラグが暫定であるウィンドウ全てを、それらの中でみたスタック順はそのまま、全体スタックの最下位とし、更に対象ウィンドウをリストアップしたウィンドウの中で最上位となるよう、ウィンドウシステム 12 にスタック変更要求を発行し、次に、リストアップしたウィンドウのスタック位置確定

フラグを確定済みに更新し、また、内部データベース全体に対してスタック位置情報を更新した上で、ステップ S 7 1 2 に進む。

ステップ S 7 1 1 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 1 1 に示す。

図 1 1 に示されるように、対象ウィンドウと同じグループ値を持つスタック位置確定フラグが暫定の通常ウィンドウが、スタック最下位に移され、同一グループ値を持つ通常ウィンドウ全体の中で対象ウィンドウのスタック最上位移動がなされているのが解る。

図 7 において、ステップ S 7 1 2 では、処理を終了する。

#### <グループ内スタック最下位移動要求>

図 1 において、アプリケーションプログラム 1 0 1, 1 0 2, 1 0 3 は、生成及びウィンドウ種類及びグループ設定したウィンドウに対して、グループ内スタック最下位移動要求をウィンドウシステム 1 2 に対して発行することができる。Xウィンドウシステムの場合、例えば、グループ内スタック最下位移動要求に `XLowerWindow()` 関数を割り当てれば良い。また、優先ウィンドウに対して、スタック変更要求として、優先ウィンドウスタック最下位移動要求をウィンドウシステム 1 2 に対して発行することができる。例えば、グループ内スタック最下位移動要求と同じ `XLowerWindow()` 関数を割り当てれば良い。

アプリケーションプログラムがX L o w e r W i n d o w ( ) を実施した場合、ウィンドウシステム12はスタック変更をせず、ウィンドウマネージャ105に対してスタック最下位移動要求イベント(C o n f i g u r e R e q u e s t ) を発行する。

スタック最下位移動要求イベントを受け取った際のウィンドウマネージャ105の動作例となるフロー図を図12に示す。

図12に示されるように、ステップS1201では、該時点で受け取ったスタック最下位移動要求イベント(C o n f i g u r e R e q u e s t ) から対象ウィンドウの識別子を獲得して、ステップS1202に進む。

ステップS1202では、対象ウィンドウの識別子で内部データベースを検索し、対応するデータが存在する場合はステップS1203へ進み、存在しない場合はステップS1212へ進む。

ステップS1203では、内部データベースにあるプロパティ獲得フラグが未獲得である全ウィンドウについて、ウィンドウシステム12に対してウィンドウ種類及びグループ値を問い合わせ、獲得できたウィンドウについては、そのウィンドウ種類及びグループ値と、プロパティ獲得済みである事を内部データベースに記憶し、ステップS1204に進む。

ステップS1204では、対象ウィンドウのウィンドウ種類をチェックし、優先ウィンドウの場合はステップS1205へ進み、通常ウィンドウの場合はステップS120

6 へ進む。

ステップ S 1 2 0 5 では、内部データベースからウィンドウ種類が優先ウィンドウであるものの内、スタック最下位のウィンドウを検索する。そして、対象ウィンドウが、そのスタック最下位のウィンドウの直下となるよう、ウィンドウシステム 1 2 にスタック変更要求を発行し、次に、内部データベース全体に対してスタック位置情報を更新した上で、ステップ S 1 2 1 2 に進む。なお、検索の結果、スタック最下位のウィンドウが対象ウィンドウ自身であった場合は、何もせずステップ S 1 2 1 2 へ進む。

ステップ S 1 2 0 5 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 1 3 に示す。

図 1 3 に示されるように、対象ウィンドウのみが、優先ウィンドウの中のスタック最下位のウィンドウの直下に移されているのが解る。

図 1 2 において、ステップ S 1 2 0 6 では、対象ウィンドウと同じグループに属する通常ウィンドウを内部データベースから検索してリストアップし、ステップ S 1 2 0 7 に進む。

ステップ S 1 2 0 7 では、前記リストアップした中にスタック位置確定フラグが確定済みであるウィンドウが存在するならばステップ S 1 2 0 8 に進み、存在しないならステップ S 1 2 0 9 に進む。

ステップ S 1 2 0 8 では、リストアップした中でスタック位置確定フラグが暫定であるウィンドウがあれば、それ

ら全てを、それらの中でみたスタック順はそのまま、リストアップした中でスタック位置確定フラグが確定済みであるウィンドウの内の最上位のウィンドウの直上に位置させ、更に対象ウィンドウをリストアップしたウィンドウの中で最下位となるよう、ウィンドウシステム 12 にスタック変更要求を発行し、次に、リストアップした中でスタック位置確定フラグが暫定であるウィンドウがあれば確定済みに更新し、また、内部データベース全体に対してスタック位置情報を更新した上で、ステップ S 1 2 1 2 に進む。

ステップ S 1 2 0 8 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 1 4 に示す。

図 1 4 に示されるように、対象ウィンドウと同じグループ値を持つスタック位置確定フラグが暫定の通常ウィンドウが、同じグループ値を持つスタック位置確定フラグが確定済みの通常ウィンドウの直上に移され、同一グループ値を持つ通常ウィンドウ全体の中で対象ウィンドウのスタック最下位移動がなされているのが解る。

図 1 2 において、ステップ S 1 2 0 9 では、内部データベースにスタック位置確定フラグが確定済みの通常ウィンドウが存在するならばステップ S 1 2 1 0 に進み、存在しないならばステップ S 1 2 1 1 に進む。

ステップ S 1 2 1 0 では、ステップ S 1 2 0 6 でリストアップした中でスタック位置確定フラグが暫定であるウィンドウ全てを、それらの中でみたスタック順はそのまま、内部データベースの中でスタック位置確定フラグが確定



済みであるスタック最上位の通常ウィンドウの直上に位置させ、更に対象ウィンドウを前記リストアップしたウィンドウの中で最下位となるよう、ウィンドウシステム 12 にスタック変更要求を発行し、次に、リストアップしたウィンドウのスタック位置確定フラグを確定済みに更新し、また、内部データベース全体に対してスタック位置情報を更新した上で、ステップ S 1 2 1 2 に進む。

ステップ S 1 2 1 0 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 1 5 に示す。

図 1 5 に示されるように、対象ウィンドウと同じグループ値を持つスタック位置確定フラグが暫定の通常ウィンドウが、スタック位置確定フラグが確定済みの通常ウィンドウ全体のスタック最上位ウィンドウの直上に移され、同一グループ値を持つウィンドウ全体の中で対象ウィンドウのスタック最下位移動がなされているのが解る。

図 1 2 において、ステップ S 1 2 1 1 では、ステップ S 1 2 0 6 でリストアップした中でスタック位置確定フラグが暫定であるウィンドウ全てを、それらの中でみたスタック順はそのまま、全体スタックの最下位とし、更に対象ウィンドウをリストアップしたウィンドウの中で最下位となるよう、ウィンドウシステム 12 にスタック変更要求を発行し、次に、リストアップしたウィンドウのスタック位置確定フラグを確定済みに更新し、また、内部データベース全体に対してスタック位置情報を更新した上で、ステップ S 1 2 1 2 に進む。

ステップ S 1 2 1 1 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 1 6 に示す。

図 1 6 に示されるように、対象ウィンドウと同じグループ値を持つスタック位置確定フラグが暫定の通常ウィンドウが、スタック最下位に移され、同一グループ値を持つ通常ウィンドウ全体の中で対象ウィンドウのスタック最下位移動がなされているのが解る。

図 1 2 において、ステップ S 1 2 1 2 では、処理を終了する。

#### <グループ単位スタック最上位移動要求>

図 1 において、アプリケーションプログラム 1 0 1 , 1 0 2 , 1 0 3 は、生成及びウィンドウ種類及びグループ設定した通常ウィンドウに対して、グループ単位スタック最上位移動要求をウィンドウシステム 1 2 に対して発行することができる。Xウィンドウシステムの場合、例えば、クライアントメッセージ機構を利用する事で容易に実現できる。今後、クライアントメッセージでウィンドウシステム 1 2 に対して発行するグループ単位スタック最上位移動要求のメッセージの事をグループ単位スタック最上位移動要求メッセージと呼ぶ事とする。そして、このグループ単位スタック最上位移動要求メッセージには、移動したいグループ（対象グループ）のグループ値が含まれているものとする。

アプリケーションプログラム 1 0 1 , 1 0 2 , 1 0 3 がグループ単位スタック最上位移動要求メッセージ（C l i

ent Message)を発行した場合、このメッセージはウィンドウシステム12経由でウィンドウマネージャ105に転送される。

グループ単位スタック最上位移動要求メッセージを受け取った際のウィンドウマネージャ105の動作例となるフローを図17に示す。

図17に示されるように、ステップS1701では、該時点で受け取ったグループ単位スタック最上位移動要求メッセージ(Client Message)から対象グループ値を獲得して、ステップS1702に進む。

ステップS1702では、内部データベースにあるプロパティ獲得フラグが未獲得である全ウィンドウについて、ウィンドウシステム12に対してウィンドウ種類及びグループ値を問い合わせ、獲得できたウィンドウについては、そのウィンドウ種類及びグループ値と、プロパティ獲得済みである事を内部データベースに記憶し、ステップS1703に進む。

ステップS1703では、対象グループ値を持つ通常ウィンドウ情報を内部データベースから検索してリストアップし、ステップS1704に進む。

ステップS1704では、内部データベースにスタック位置確定情報が確定済み且つグループ値情報が対象グループ値とは異なる通常ウィンドウが存在すればステップS1705に進み、存在しなければステップS1706に進む。

ステップS1705では、リストアップしたウィンドウ

が、それらの中でのスタック順はそのまま、内部データベースの中でスタック位置確定フラグが確定済み且つグループ値情報が対象グループ値とは異なる通常ウィンドウの内でスタック最上位のウィンドウの直上となるよう、ウィンドウシステム 12 にスタック変更要求を発行し、次に、リストアップした中でスタック位置確定フラグが暫定であるウィンドウがあれば確定済みに更新し、また、内部データベース全体に対してスタック位置情報を更新した上で、ステップ S 1 7 0 7 に進む。

ステップ S 1 7 0 5 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 1 8 に示す。

図 1 8 に示されるように、対象グループ値を持つ通常ウィンドウが、スタック位置確定フラグが確定済み且つグループ値情報が対象グループ値とは異なる通常ウィンドウ全体のスタック最上位ウィンドウの直上に移されているのが解る。

図 1 7 におけるステップ S 1 7 0 6 では、リストアップしたウィンドウが、それらの中でのスタック順はそのまま、全体スタックの最下位となるよう、ウィンドウシステム 12 にスタック変更要求を発行し、次に、リストアップした中でスタック位置確定フラグが暫定であるウィンドウがあれば確定済みに更新し、また、内部データベース全体に対してスタック位置情報を更新した上で、ステップ S 1 7 0 7 に進む。

ステップ S 1 7 0 6 のスタック変更要求によってスタック

クがどのように変化するかを示す具体例となる模式図を図 19 に示す。

図 19 に示されるように、対象グループ値を持つ通常ウィンドウが、全体スタックの最下位に移されているのが解る。

図 17 において、ステップ S 1707 では、処理を終了する。

#### ＜グループ単位スタック最下位移動要求＞

図 1 において、アプリケーションプログラム 101, 102, 103 は、生成及びウィンドウ種類及びグループ設定した通常ウィンドウに対して、グループ単位スタック最下位移動要求をウィンドウシステム 12 に対して発行することができる。Xウィンドウシステムの場合、例えば、クライアントメッセージ機構を利用する事で容易に実現できる。今後、クライアントメッセージでウィンドウシステム 12 に対して発行するグループ単位スタック最下位移動要求のメッセージの事をグループ単位スタック最下位移動要求メッセージと呼ぶ事とする。そして、このグループ単位スタック最下位移動要求メッセージには、移動したいグループ（対象グループ）のグループ値が含まれているものとする。

アプリケーションプログラム 101, 102, 103 がグループ単位スタック最下位移動要求メッセージ（Client Message）を発行した場合、このメッセージはウィンドウシステム 12 経由でウィンドウマネージャ 105 に転送される。

グループ単位スタック最下位移動要求メッセージを受け取った際のウィンドウマネージャ 105 の動作例となるフローを図 20 に示す。

図 20 に示されるように、ステップ S 2001 では、該時点で受け取ったグループ単位スタック最下位移動要求メッセージ (Client Message) から対象グループ値を獲得して、ステップ S 2002 に進む。

ステップ S 2002 では、内部データベースにあるプロパティ獲得フラグが未獲得である全ウィンドウについて、ウィンドウシステム 12 に対してウィンドウ種類及びグループ値を問い合わせ、獲得できたウィンドウについては、そのウィンドウ種類及びグループ値と、プロパティ獲得済みである事を内部データベースに記憶し、ステップ S 2003 に進む。

ステップ S 2003 では、対象グループ値を持つ通常ウィンドウを内部データベースから検索してリストアップし、ステップ S 2004 に進む。

ステップ S 2004 では、リストアップしたウィンドウが、それらの中でのスタック順はそのままで、全体スタックの最下位となるよう、ウィンドウシステム 12 にスタック変更要求を発行し、次に、リストアップした中でスタック位置確定フラグが暫定であるウィンドウがあれば確定済みに更新し、また、内部データベース全体に対してスタック位置情報を更新した上で、ステップ S 2005 に進む。

ステップ S 2004 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図

21に示す。

図21に示されるように、対象グループ値を持つ通常ウィンドウが、全体スタックの最下位に移されているのが解る。

図20におけるステップS2005では、処理を終了する。

#### <破棄要求>

図1において、アプリケーションプログラム101, 102, 103は、生成したウィンドウに対して破棄要求を発行する。

ウィンドウシステム12は、破棄要求を受け取ると、該当するウィンドウを管理から外して削除する。そして、ウィンドウマネージャ105にウィンドウ破棄通知イベント(DestroyNotify)を発行する。

ウィンドウマネージャ105は、ウィンドウ破棄通知イベント(DestroyNotify)を受け取ると、そこからウィンドウ識別子を抽出して、該当するウィンドウ情報を内部データベースから検索し削除した後、内部データベース全体に対して必要に応じてスタック位置を更新し、処理を終了する。

以上のことにより、各通常ウィンドウはそのウィンドウの表示までに、ウィンドウのスタック順がグループ毎に纏められるので、最上位のグループの各ウィンドウは、他のグループのウィンドウの下に重ねられて表示される事がなくなる。また、ウィンドウ管理プログラムが、スタック変更要求を受け取る事を切っ掛けにして、スタック変更対象

となるグループに属する通常ウィンドウ全てを、その中で  
のスタック順は保持したままで纏めた上でスタック変更を  
実施するので、アプリケーションプログラムは、自グルー  
プのウィンドウのスタック順を把握し易くなる。また、ウ  
ィンドウ管理プログラムにグループ内のスタック最上位移  
動とグループ内のスタック最下位移動の機能を設ける事で  
、グループ内で通常ウィンドウのスタック順を操作できる  
ので、各アプリケーションプログラムは、他のアプリケー  
ションプログラムの通常ウィンドウ表示を気にする事無く  
、スタック変更を行う事ができる。また、ウィンドウ管理  
プログラムにグループ単位のスタック最上位移動と最下位  
移動の機能を設ける事で、例えば、グループをアプリケー  
ションプログラム単位で振り分けた場合だと、そのアプリ  
ケーションプログラムに属するウィンドウのメンバーを知  
らなくても、スタック最前面（最上位）にそのアプリケー  
ションプログラムに属する全通常ウィンドウを移動させる  
事が可能となる。また、優先ウィンドウを設ける事で、グ  
ループ単位のスタック順に関係無く、特定のウィンドウを  
最優先で表示することのできるサービスを提供することが  
できる。

なお、本実施の形態では、ウィンドウマネージャにおけ  
るウィンドウ種類及びグループ値取得のタイミングをマッ  
プ要求イベント（MapRequest）受信時及びスタ  
ック変更要求受信時としたが、これに限定されるものでは  
なく、アプリケーションプログラムがウィンドウ種類及び  
グループ設定をしたタイミングで取得して内部データベー



スに記憶しておいても良く、その方がむしろ処理効率が良い。更に、このタイミングで、スタック順をグループ毎に纏めても構わない。また、ウィンドウ種類及びグループ値の変更を認めても良いし、認めなくても良い。認める場合は、ウィンドウ種類及びグループ設定のタイミングで、ウィンドウ種類が優先ウィンドウから通常ウィンドウに変更された場合は、スタック位置確定フラグを暫定に変更する。逆にウィンドウ種類が通常ウィンドウから優先ウィンドウに変更された場合は、該時点のスタック最上位に該ウィンドウを位置させるようにスタック変更を行う。また、ウィンドウ種類に変更が無く、グループ値が変更となった場合は、スタック位置確定フラグを暫定に変更する。そして、内部データベースを更新すれば良い。また、認めない場合は、ウィンドウ種類及びグループ設定のタイミングで、プロパティ獲得フラグをチェックして、獲得済みであれば、該設定を無視すれば良い。

また、ウィンドウ種類として優先ウィンドウと通常ウィンドウの2種類を制御する例を示したが、通常ウィンドウの1種類だけであっても容易に実施できる事は言うまでも無い。

また、ウィンドウマネージャがルートウィンドウの子ウィンドウであるトップレベルウィンドウのスタック順を制御する場合を例に示したが、トップレベルウィンドウの子孫ウィンドウに対しても実施する事ができる事は言うまでも無い。

また、ウィンドウ管理プログラムをXウィンドウシステ

ムとウィンドウマネージャとして、アプリケーションプログラムとのインターフェースにウィンドウ生成要求、マップ要求を用いたが、これに限定されるものではなく、表示要求に相当するインターフェースさえあれば良く、その場合、ウィンドウ管理プログラムは、ウィンドウ表示要求受信時に、本実施の形態で示したウィンドウ生成要求受信時の処理とマップ要求受信時の処理を纏めて行えば良い。

また、ウィンドウのスタック順をグループ毎に一連とする際、単にウィンドウのスタック順を変更するだけであったが、例えば、ウィンドウ管理プログラムが、グループ毎に代表ウィンドウを1枚生成し、ウィンドウのスタック順をグループ毎に一連とする際に、前記ウィンドウを前記代表ウィンドウの子ウィンドウとするようにして制御することも容易である。そうすることによって、グループ単位のスタック移動やグループ単位のウィンドウの表示／非表示の切り替えを代表ウィンドウに対する制御で実施できる分、グループ内のウィンドウが多いほど、処理負荷を軽減することができる。代表ウィンドウ使用下でのスタックの様子を示す例となる模式図を図22に示す。

また、最上位グループよりスタック下位のウィンドウ表示について、ウィンドウ管理プログラムでは非表示とするような制御はしていないが、非表示とする制御を行う事も容易に実現できる。その場合、ウィンドウ情報に表示／非表示を示すマップフラグを追加して、このマップフラグによりアプリケーションプログラムの表示要求状態を記憶しておき、該ウィンドウの所属するグループが通常ウィンド

ウの中でスタック最上位となった時、前記マップフラグをチェックし、表示となっていればマップ処理を行い、非表示となっていればアンマップのままとする。逆に、該ウィンドウの所属するグループが通常ウィンドウの中でスタック最上位でなくなった際には、前記マップフラグが表示となっていればアンマップ処理を行い、非表示となっていれば何もしない。これにより、通常ウィンドウの中の最上位グループよりスタック下位のグループが何を表示しようとしても表示装置に表示されないので、最上位グループのウィンドウで覆われない部分にスタック下位のグループのウィンドウが中途半端に映り込む心配が無い。

また、本実施形態では、通常ウィンドウの最上位グループに属する全ウィンドウの中のスタック最下位のウィンドウの直下に、ウィンドウ管理プログラムが生成及びマップする特定のウィンドウを位置させることは特にしなかったが、位置させることも容易に実現できる。これにより、例えば、前記特定のウィンドウを表示装置のスクリーン一杯に表示しておけば、前記特定のウィンドウのスタック下位のグループが何を表示しようとしても、表示装置に表示されないので、最上位グループのウィンドウで覆われない部分にスタック下位のグループのウィンドウが中途半端に映り込む心配が無い。前記特定のウィンドウを使用した場合のスタックの様子を示す例となる模式図を図23に示し、また、スクリーンに表示されている例となる模式図を図24に示す。

また、アプリケーションプログラムが3つの場合を例に

示したが、これに限定されるものではなく、1つ以上であれば良い。

また、各アプリケーションプログラムに1つのグループを割り当てた場合を例に示したが、これに限定されるものではなく、1つのアプリケーションに複数のグループを割り当てても良く、また、複数のアプリケーションに1つのグループを割り当てても構わない。

また、ウィンドウ管理プログラムをXウィンドウシステムとウィンドウマネージャとで構成し、両者間でイベントや各種要求を送受信し合う例を示したが、これに限定されるものではなく、Xウィンドウシステムである必要は無いし、また、2つに分けてイベントや各種要求を送受信する構成である必要もなく、一体化して実施する場合は、イベントやメッセージのインターフェースを関数ベースに変える事で本実施の形態を容易に応用できる。

また、通常ウィンドウの最上位グループに属する全ウィンドウの中のスタック最上位のウィンドウの直上に、ウィンドウ管理プログラムが生成する特定のウィンドウを位置させることは特にしていないが、位置させることも容易に実現できる。これにより、最上位グループを別のグループに入れ替える際、Xウィンドウシステムの場合、`XRestackWindows()` 関数を使用する事となるが、その特定のウィンドウをウィンドウ群のスタック移動先を示すために使用できる。もし、最上位グループのウィンドウよりスタック上位にウィンドウが存在しなかった場合、存在する全てのウィンドウの並び順を決め、前記 `XRestackWindows()`

t a c k W i n d o w s ( ) の引数に代入せねばならないが、前記特定のウィンドウがあれば、前記特定のウィンドウと前記ウィンドウ群を引数に代入するだけで済む。以下、その理由を簡単に説明する。X R e s t a c k W i n d o w s ( ) 関数でスタック順を変更するには2通りの方法がある。1つ目の方法は、引数として、既存のウィンドウの全てをスタック順に並べる全体指定による方法である。2つ目の方法は、引数として、スタック順を変えない1つのウィンドウと、その下位のウィンドウの内のスタック順を変える部分までを指定する部分指定による方法である。例えば、スタック順上位から、A, B, C, D, Eの5つのウィンドウが存在していて、このスタック順をA, C, B, D, Eに変更する場合、全体指定の引数としては(A, C, B, D, E)を、部分指定の引数として(A, C, B)を代入すればよい。このように、部分指定は、全体指定に比べて、引数に指定すべきウィンドウの数が少なく済むというメリットがある。上記のように、通常ウィンドウの最上位グループに属する全ウィンドウの中のスタック最上位のウィンドウの直上に、ウィンドウ管理プログラムが生成する特定のウィンドウを位置させることにより、その特定のウィンドウを、部分指定の引数における「スタック順を変えない1つのウィンドウ」として利用することができるため、最上位グループを別のグループに入れ替える際にも部分指定によるスタック順の変更が可能となる。なお、部分指定の他のメリットとして、X R e s t a c k W i n d o w s ( ) 関数の実行が成功し易くなることがある。

。なぜなら、ウィンドウシステムとウィンドウマネージャとは非同期に動作しており、アプリケーションプログラムからの要求に基づくウィンドウ生成・破棄は、マップ（ウィンドウ表示）とは異なり、アプリケーションプログラムとウィンドウシステムとの間で勝手に実施されるため、ウィンドウマネージャが管理している、ウィンドウの有無に関する情報は、最新の情報とは異なっている場合があるからである。この場合、全体指定では全ウィンドウのスタック順を指定する必要があるため、スタック順の変更に失敗してしまうのである。

また、Xウィンドウシステムを用いる場合、ウィンドウマネージャは、スタックを操作するために `XRestackWindows()` 関数を使用する。`XRestackWindows()` 関数の引数には、ウィンドウ群とそのウィンドウ群の移動先を示すウィンドウを指定するが、それらの何れかが破棄されていた場合、スタック操作が意図した通りに実施されない事がある。そのための対策として、例えば、ウィンドウマネージャは、ウィンドウ破棄通知イベント（`DestroyNotify`）を受け取った際に、該当するウィンドウ情報を内部データベースから検索し削除し、内部データベース全体に対して必要に応じてスタック位置を更新した後、次に、ウィンドウシステムの該時点でのスタックツリーを問い合わせで獲得し、ウィンドウマネージャが認識していない新規のウィンドウは対象から外し、ウィンドウマネージャのスタックツリーとの整合性をチェックし、ウィンドウマネージャの認識と異なって

いる場合は、ウィンドウマネージャのスタックツリーに合わせ込むスタック変更要求を発行し、認識が合うまで、ウィンドウシステムの該時点でのスタックツリー問い合わせ、整合性チェック、スタック変更要求の発行を繰り返すようにしても良い。これにより、ウィンドウシステムがスタック変更処理に失敗しても、ウィンドウシステム間とのスタック認識の整合性を確保でき、処理が破綻する事がなくなる。

また、ウィンドウマネージャが、X R e s t a c k W i n d o w s ( ) を実行した際にフラグを立て、ウィンドウ破棄通知イベント ( D e s t r o y N o t i f y ) を受け取った際に、前記フラグが立っている場合にのみ、前記フラグを下げると共にスタックツリーの整合性のための処理を行うようにしても良い。これにより、スタックツリーの整合性のための処理の実行頻度を軽減できる。

また、コンピュータ 7 は、家庭用コンピュータや携帯電話、P D A ( P e r s o n a l D i g i t a l A s s i s t a n c e ) 、セットトップボックス、デジタルスチルカメラ、カメラ一体型 V T R 等、プログラムが動作するための C P U とメモリを実装して表示装置に複数のウィンドウを表示しようとする装置であれば何でも良い事は言うまでもない。

また、表示装置 1 1 をコンピュータ 7 の一部としたが、物理上で一体化している事を指すものではなく、コンピュータ 7 と有線／無線を問わず、何らかの形で接続されていさえすれば良い。

また、本実施の形態のウィンドウスタック制御方法を実現するためのウィンドウ管理プログラムは、ROMやディスク等に記憶してコンピュータに供給する事ができるほか、ネットワークを介してコンピュータに供給する事が可能である事は言うまでもない。

(実施の形態 2)

実施の形態 2 のウィンドウスタック制御方法は、実施の形態 1 を変形したものであり、実施の形態 1 と実施の形態 2 の違いは、マップ要求イベント (Map Request) を受け取った際のウィンドウマネージャ 105 の動作と、アプリケーションプログラムがウィンドウ種類及びグループ値を設定するタイミングでウィンドウマネージャ 105 はスタック順をグループ毎に纏めてはいけないことだけである。よって、本実施例では、ウィンドウマネージャ 105 のマップ要求イベント (Map Request) を受け取った際のウィンドウマネージャ 105 の動作についてのみ説明する。

マップ要求イベント (Map Request) を受け取った際のウィンドウマネージャ 105 の動作例となるフロー図を図 25 に示す。

図 25 に示されるように、ステップ S2501 では、該時点で受け取ったマップ要求イベント (Map Request) からマップ対象ウィンドウの識別子を獲得して、ステップ S2502 に進む。

ステップ S2502 では、マップ対象ウィンドウの識別子で内部データベースを検索し、対応するウィンドウ情報



が存在する場合はステップ S 2 5 0 3 へ進み、存在しない場合はステップ S 2 5 1 5 へ進む。

ステップ S 2 5 0 3 では、内部データベースで対象ウィンドウのスタック位置確定フラグが確定済みかどうかをチェックし、確定済みの場合はステップ S 2 5 1 4 に進み、暫定の場合はステップ S 2 5 0 4 に進む。

ステップ S 2 5 0 4 では、対象ウィンドウのプロパティ獲得フラグをチェックし、獲得済みの場合はステップ S 2 5 0 6 に進み、未獲得の場合はステップ S 2 5 0 5 に進む。

ステップ S 2 5 0 5 では、対象ウィンドウのウィンドウ種類及びグループ値をウィンドウシステムに問い合わせして獲得し、内部データベースに格納し、ステップ S 2 5 0 6 に進む。

ステップ S 2 5 0 6 では、対象ウィンドウのウィンドウ種類をチェックし、優先ウィンドウの場合は、ステップ S 2 5 1 4 に進み、通常ウィンドウの場合は、ステップ S 2 5 0 7 に進む。

ステップ S 2 5 0 7 では、内部データベースにあるプロパティ獲得フラグが未獲得である全ウィンドウについて、ウィンドウシステム 1 2 に対してウィンドウ種類及びグループ値を問い合わせ、獲得できたウィンドウについては、そのウィンドウ種類及びグループ値と、プロパティ獲得済みである事を内部データベースに記憶し、ステップ S 2 5 0 8 に進む。

ステップ S 2 5 0 8 では、対象ウィンドウと同じグルー

ブに属する通常ウィンドウを内部データベースから検索してリストアップし、ステップ S 2 5 0 9 に進む。

ステップ S 2 5 0 9 では、前記リストアップした中にスタック位置確定フラグが確定済みであるウィンドウが存在するならばステップ S 2 5 1 0 に進み、存在しないならばステップ S 2 5 1 1 に進む。

ステップ S 2 5 1 0 では、リストアップした中でスタック位置確定フラグが暫定であるウィンドウ全てを、それらの中でみたスタック順はそのまま、リストアップした中でスタック位置確定フラグが確定済みであるウィンドウの内の最上位のウィンドウの直上となるよう、ウィンドウシステム 1 2 にスタック変更要求を発行し、ステップ S 2 5 1 4 に進む。

ステップ S 2 5 1 0 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 2 6 に示す。

図 2 6 に示されるように、マップ対象ウィンドウ及びマップ対象ウィンドウと同じグループ値を持つスタック位置確定フラグが暫定の通常ウィンドウが、前記グループ値を持つスタック位置確定フラグが確定済みの通常ウィンドウの直上に移されているのが解る。

図 2 5 において、ステップ S 2 5 1 1 では、内部データベースにスタック位置確定フラグが確定済みの通常ウィンドウが存在するならばステップ S 2 5 1 2 に進み、存在しないならばステップ S 2 5 1 3 に進む。

ステップ S 2 5 1 2 では、ステップ S 2 5 0 8 でリスト

アップした中でスタック位置確定フラグが暫定である通常ウィンドウ全てを、それらの中でみたスタック順はそのまま、内部データベースの中でスタック位置確定フラグが確定済みであるスタック最上位の通常ウィンドウの直上となるよう、ウィンドウシステム 12 にスタック変更要求を発行し、ステップ S 2 5 1 4 に進む。

ステップ S 2 5 1 2 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 2 7 に示す。

図 2 7 に示されるように、マップ対象ウィンドウ及びマップ対象ウィンドウと同じグループ値を持つスタック位置確定フラグが暫定の通常ウィンドウが、スタック位置確定フラグが確定済みの通常ウィンドウ全体のスタック最上位ウィンドウの直上に移されているのが解る。

図 2 5 において、ステップ S 2 5 1 3 では、ステップ S 2 5 0 8 でリストアップした中でスタック位置確定フラグが暫定である通常ウィンドウ全てを、それらの中でみたスタック順はそのまま、全体スタックの最下位となるよう、ウィンドウシステム 12 にスタック変更要求を発行し、ステップ S 2 5 1 4 に進む。

ステップ S 2 5 1 3 のスタック変更要求によってスタックがどのように変化するかを示す具体例となる模式図を図 2 8 に示す。

図 2 8 に示されるように、マップ対象ウィンドウ及びマップ対象ウィンドウと同じグループ値を持つスタック位置確定フラグが暫定の通常ウィンドウが、スタック最下位に

移されているのが解る。

図 2 5 において、ステップ S 2 5 1 4 では、ウィンドウシステム 1 2 に対してマップ対象ウィンドウのマップ要求を発行し、内部データベースの対象ウィンドウのウィンドウ情報の内、スタック位置確定フラグが未確定を示している場合は確定済みに更新し、また、内部データベース全体に対して必要に応じてスタック位置を更新した上で、ステップ S 2 5 1 5 に進む。

ステップ S 2 5 1 5 では、処理を終了する。

以上のことにより、ウィンドウ管理プログラムが、マップ対象ウィンドウと同じグループ値を持つスタック位置が暫定の通常ウィンドウ群を、その通常ウィンドウ群の中でのスタック上下関係は保持しつつスタック位置を移してグループ毎に纏めるため、マップによって、同一グループ内でみたスタック順は変更されず、アプリケーションの同一グループ内のスタック順管理の負担が軽減される。更に、実施の形態 1 で示したように、スタック変更要求受信時にも、対象ウィンドウと同じグループ値を持つスタック位置が暫定の通常ウィンドウ群を、その通常ウィンドウ群の中でのスタック上下関係は保持しつつグループ毎に纏める事で、アプリケーションプログラムは、自グループに属するウィンドウのスタック順をウィンドウの生成順を基本として管理する事ができる。これは、Xウィンドウシステムの基本と一致しており、アプリケーションプログラムは、自グループの通常ウィンドウについて、一般的なスタック順決定ルールである生成順ベースで重なりを管理すれば良く

なる。

なお、本実施の形態では、マップ要求イベント（Map Request）受信時に、ウィンドウマネージャのウィンドウのグループ値取得を行っているが、これに限定されるものではなく、実施の形態 1 と同様、アプリケーションプログラムがウィンドウ種類及びグループ設定をしたタイミングでグループ値を取得して内部データベースにウィンドウ種類とグループ値とプロパティ獲得済みである事を記憶しておいても良く、その方がむしろ処理効率が良い。しかし、実施の形態 1 と違い、前記タイミングで、スタック順をグループ毎に纏めてはいけない。

#### 産業上の利用可能性

本発明は、表示装置に複数のウィンドウを表示するのに好適であり、特に、画面が小さく、結果として、ウィンドウの重なり易い携帯電話や P D A 等におけるウィンドウの管理に好適である。

## 請求の範囲

1. 1つ以上のアプリケーションプログラムに基づいて表示装置に表示される複数のウィンドウの重ね合わせを管理するためのウィンドウスタック制御方法であって、

アプリケーションプログラムからウィンドウのグループの指定を受けるステップと、

アプリケーションプログラムからウィンドウの表示要求を受けるステップと、

表示要求を受けて当該ウィンドウの表示を行う際に、当該ウィンドウのスタック順をグループ毎に一連となるように纏めるステップとを備えたウィンドウスタック制御方法。

2. 前記纏めるステップは、グループ毎に纏めていない第1のウィンドウの表示要求をアプリケーションプログラムから受け取った際に、グループ毎に纏めていないウィンドウの中で前記第1のウィンドウと同じグループに属する第1のウィンドウ群のスタック順が、該第1のウィンドウ群の中でのスタック順関係は保持したまま、既にグループ毎に纏められたウィンドウの中で前記第1のウィンドウと同じグループに属する第2のウィンドウ群のスタック順と一連となるようにすることを特徴とする、請求項1に記載のウィンドウスタック制御方法。

3. 前記ウィンドウスタック制御方法は、グループ毎に代表ウィンドウを1枚生成するステップをさらに備え、

前記纏めるステップは、ウィンドウのスタック順をグル

ープ毎に一連とする際に、当該ウィンドウを前記代表ウィンドウの子ウィンドウとする事を特徴とする、請求項 1 または 2 に記載のウィンドウスタック制御方法。

4. アプリケーションプログラムからグループ内でのウィンドウスタックの最上位移動要求および最下位移動要求を受けるステップと、

前記最上位移動要求または前記最下位移動要求を受けて、グループ内でのスタック変更を行うステップとをさらに備えた、請求項 1 から 3 のいずれかに記載のウィンドウスタック制御方法。

5. 前記スタック変更を行うステップは、前記第 1 のウィンドウに対するグループ内でのウィンドウスタックの最上位移動要求および最下位移動要求をアプリケーションプログラムから受け取った際に、グループ毎に纏めていないウィンドウの中で前記第 1 のウィンドウと同じグループに属する第 1 のウィンドウ群のスタック順が、前記第 1 のウィンドウ群の中でのスタック順関係は保持したまま、既にグループ毎に纏められたウィンドウの中で前記第 1 のウィンドウと同じグループに属する第 2 のウィンドウ群のスタック順と一連となるようにスタック変更を行うことを特徴とする、請求項 4 に記載のウィンドウスタック制御方法。

6. アプリケーションプログラムからグループ単位でのウィンドウスタックの最上位移動要求および最下位移動要求を受けるステップと、

前記最上位移動要求または前記最下位移動要求を受けて、グループ単位でのスタック変更を行うステップとをさら

に備えた、請求項 1 から 3 のいずれかに記載のウィンドウスタック制御方法。

7. 前記スタック変更を行うステップは、第 1 のグループに対するグループ単位でのウィンドウスタックの最上位移動要求および最下位移動要求をアプリケーションプログラムから受け取った際に、グループ毎に纏めていないウィンドウの中で前記第 1 のグループに属する第 1 のウィンドウ群のスタック順が、前記第 1 のウィンドウ群の中でのスタック順関係は保持したまま、既にグループ毎に纏められたウィンドウの中で前記第 1 のグループに属する第 2 のウィンドウ群のスタック順と一連となるようにスタック変更を行うことを特徴とする、請求項 5 に記載のウィンドウスタック制御方法。

8. アプリケーションプログラムによって優先ウィンドウとして指定されたウィンドウをいずれのグループにも所属させず、かつ当該優先ウィンドウを、表示装置に表示された、いずれかのグループに所属する全てのウィンドウより常にスタック上位に位置させることを特徴とする、請求項 1 に記載のウィンドウスタック制御方法。

9. 最上位グループに属するウィンドウよりスタック下位のウィンドウを常に非表示状態とすることを特徴とする、請求項 1 に記載のウィンドウスタック制御方法。

10. 最上位グループに属するウィンドウの中のスタック最下位のウィンドウの直下に、特定のウィンドウを位置させることを特徴とする、請求項 1 に記載のウィンドウスタック制御方法。



1 1. 前記ウィンドウスタック制御方法は、Xウィンドウシステムとウィンドウマネージャによって複数のウィンドウの重ね合わせを管理するものであって、

最上位グループの直上に特定のウィンドウを位置させることを特徴とする、請求項 4 または 6 に記載のウィンドウスタック制御方法。

1 2. 前記ウィンドウスタック制御方法は、Xウィンドウシステムとウィンドウマネージャによって複数のウィンドウの重ね合わせを管理するものであって、

ウィンドウマネージャが、ウィンドウ破棄通知受信時に、ウィンドウシステム間とでスタック認識の整合性を確認し、異なっている場合は、ウィンドウシステムのスタック認識をウィンドウマネージャのスタック認識に合わせ込む処理を行うことを特徴とする、請求項 1 から 3 のいずれかに記載のウィンドウスタック制御方法。

1 3. 前記ウィンドウスタック制御方法は、Xウィンドウシステムとウィンドウマネージャによって複数のウィンドウの重ね合わせを管理するものであって、

ウィンドウマネージャが、ウィンドウシステムにスタック変更を要求する際にフラグを立てる一方で、ウィンドウ破棄通知受信時に、前記フラグが立っている時のみ、ウィンドウシステム間とでスタック認識の整合性を確認し、異なっている場合は、ウィンドウシステムのスタック認識をウィンドウマネージャのスタック認識に合わせ込む処理を行い、前記フラグを下げることを特徴とする、請求項 1 から 3 のいずれかに記載のウィンドウスタック制御方法。

14. 前記纏めるステップは、グループ毎に纏めていない第1のウィンドウの表示要求をアプリケーションプログラムから受け取った際に、既にグループ毎に纏められたウィンドウの中で前記第1のウィンドウと同じグループに属する第2のウィンドウ群のスタック順と一連となるようにすることを特徴とする、請求項1に記載のウィンドウスタック制御方法。

15. 1つ以上のアプリケーションプログラムに基づいて表示装置に表示される複数のウィンドウの重ね合わせを管理するためのウィンドウ管理プログラムであって、コンピュータに、

アプリケーションプログラムからウィンドウのグループの指定を受けるステップと、

アプリケーションプログラムからウィンドウの表示要求を受けるステップと、

表示要求を受けた際に該ウィンドウのスタック順をグループ毎に一連となるように纏めるステップとを実行させるためのウィンドウ管理プログラム。

16. 表示装置に複数のウィンドウを表示する際のウィンドウの重ね合わせを管理するウィンドウ管理装置であって、

1つ以上のウィンドウを前記表示装置に表示する1つ以上のアプリケーションプログラムと、

前記1つ以上のアプリケーションプログラムが表示するウィンドウの重ね合わせを管理するウィンドウ管理プログラムと、

前記アプリケーションプログラムおよび前記ウィンドウ管理プログラムを実行する処理部とを備え、

前記アプリケーションプログラムは、前記ウィンドウ管理プログラムに対してウィンドウのグループを指定するものであり、

前記ウィンドウ管理プログラムは、前記アプリケーションプログラムからウィンドウの表示要求を受けて当該ウィンドウの表示を行う際に、当該ウィンドウのスタック順をグループ毎に一連とする制御を行うものであることを特徴とする、ウィンドウ管理装置。

## 補正書の請求の範囲

[2004年9月3日(03.09.2004)国際事務局受理:出願当初の請求の範囲2は取り下げられた;出願当初の請求の範囲1は補正された;他の請求の範囲は変更なし。(1頁)]

1. (補正後) 1つ以上のアプリケーションプログラムに基づいて表示装置に表示される複数のウィンドウの重ね合わせを管理するためのウィンドウスタック制御方法であって、

アプリケーションプログラムからウィンドウの新規作成要求を受けるステップと、

前記アプリケーションプログラムから前記ウィンドウのグループの指定を受けるステップと、

前記アプリケーションプログラムから前記ウィンドウの表示要求を受けるステップと、

前記表示要求を受けて当該ウィンドウの表示を行う際に、当該ウィンドウのスタック順をグループ毎に一連となるように纏めるステップと、を備え、

前記纏めるステップは、

既に一連となるように纏められたウィンドウ群におけるグループ間のスタック順関係は保持したまま、前記ウィンドウのスタック順を前記ウィンドウの属するグループと一連となるように制御する

ことを特徴とするウィンドウスタック制御方法。

2. (削除)

3. 前記ウィンドウスタック制御方法は、グループ毎に代表ウィンドウを1枚生成するステップをさらに備え、

前記纏めるステップは、ウィンドウのスタック順をグル

図 1

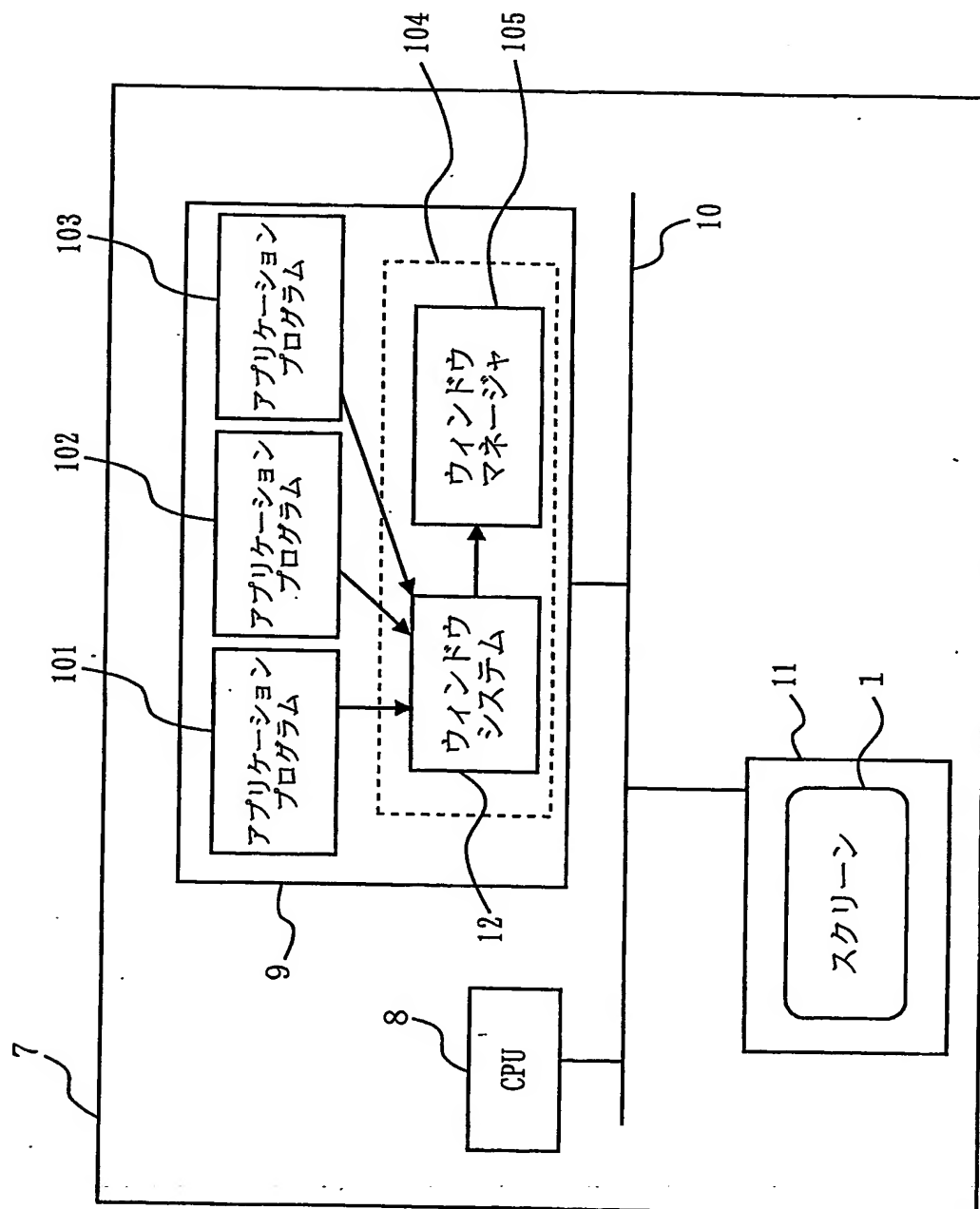
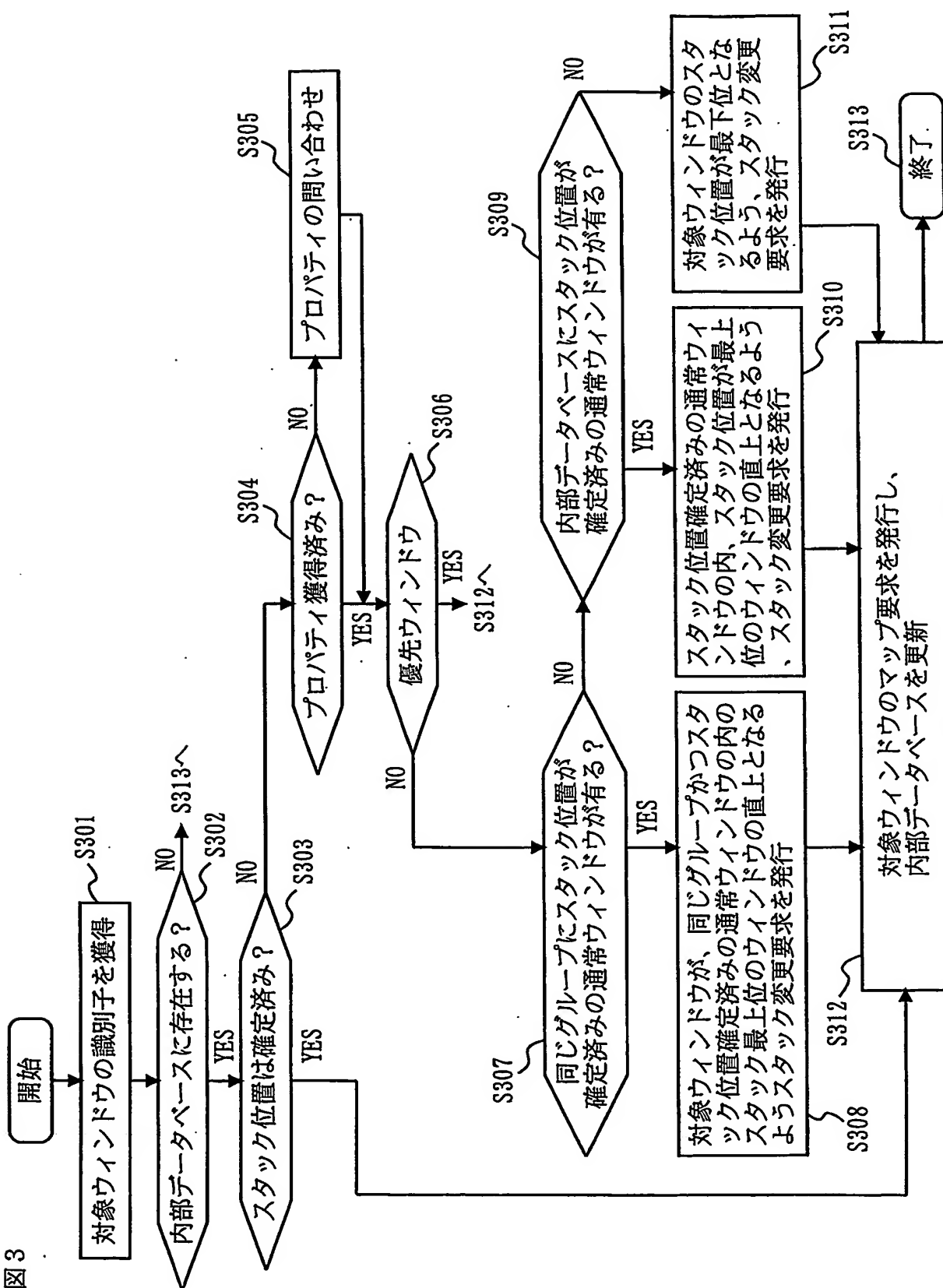


図 2

情報名	内容
ウィンドウ識別子	識別子
スタック位置確定フラグ	暫定／確定済み
スタック位置情報	位置情報
プロパティ獲得フラグ	未獲得／獲得済み
ウィンドウ種類	優先ウィンドウ／通常ウィンドウ
グループ値	グループの値

図 3



4. ☒ 4. ☐

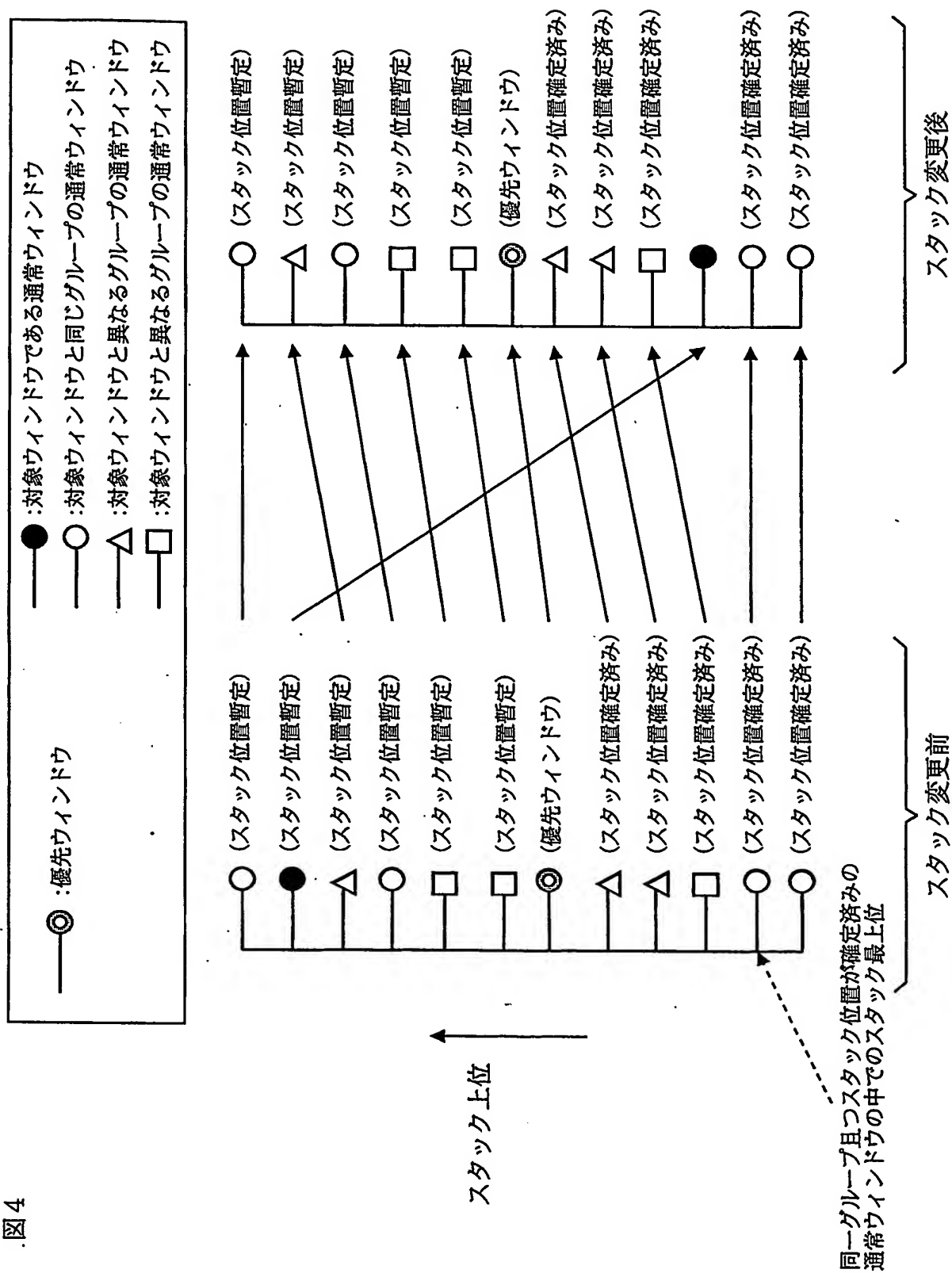




図 5

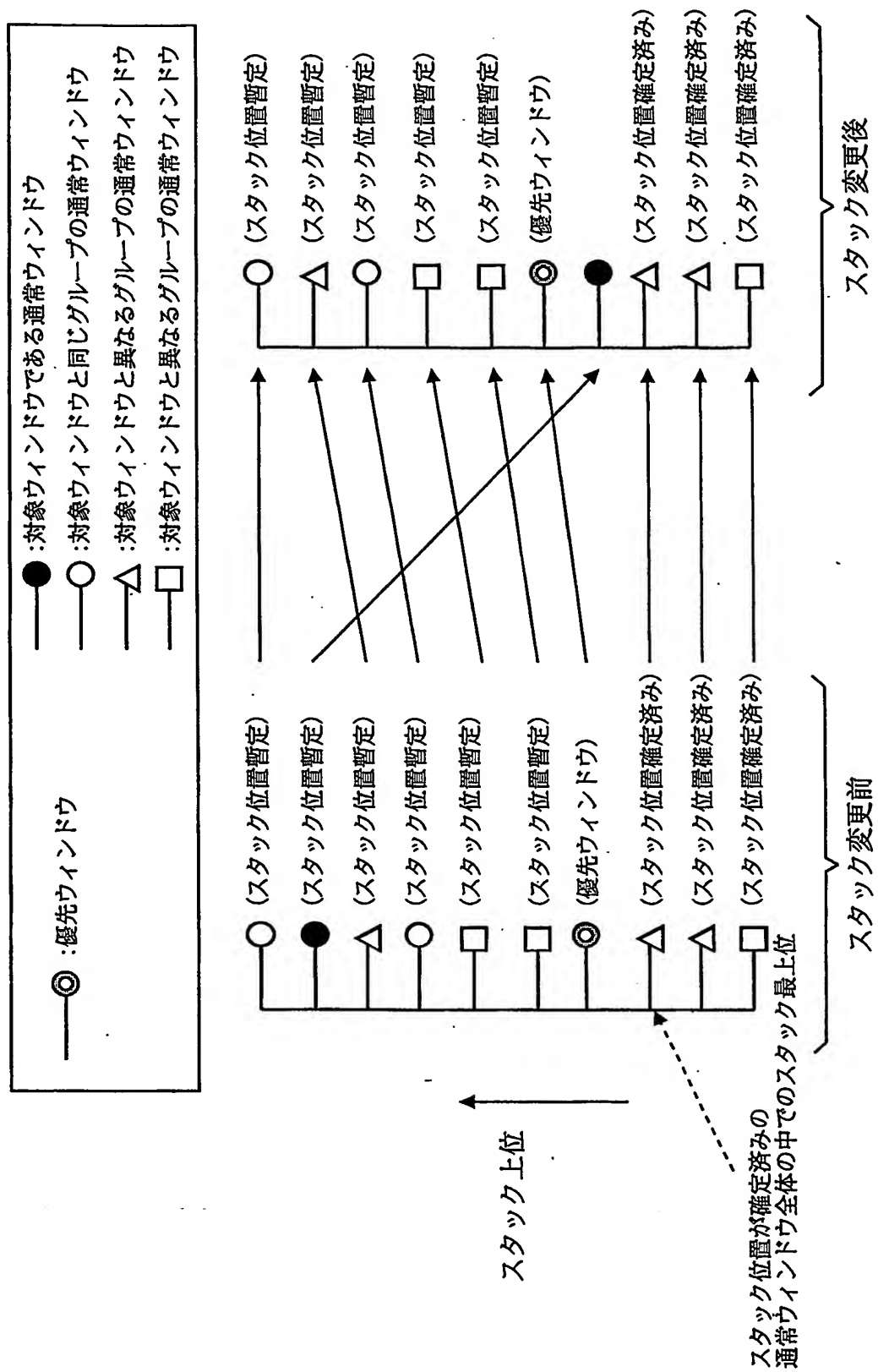


図 6

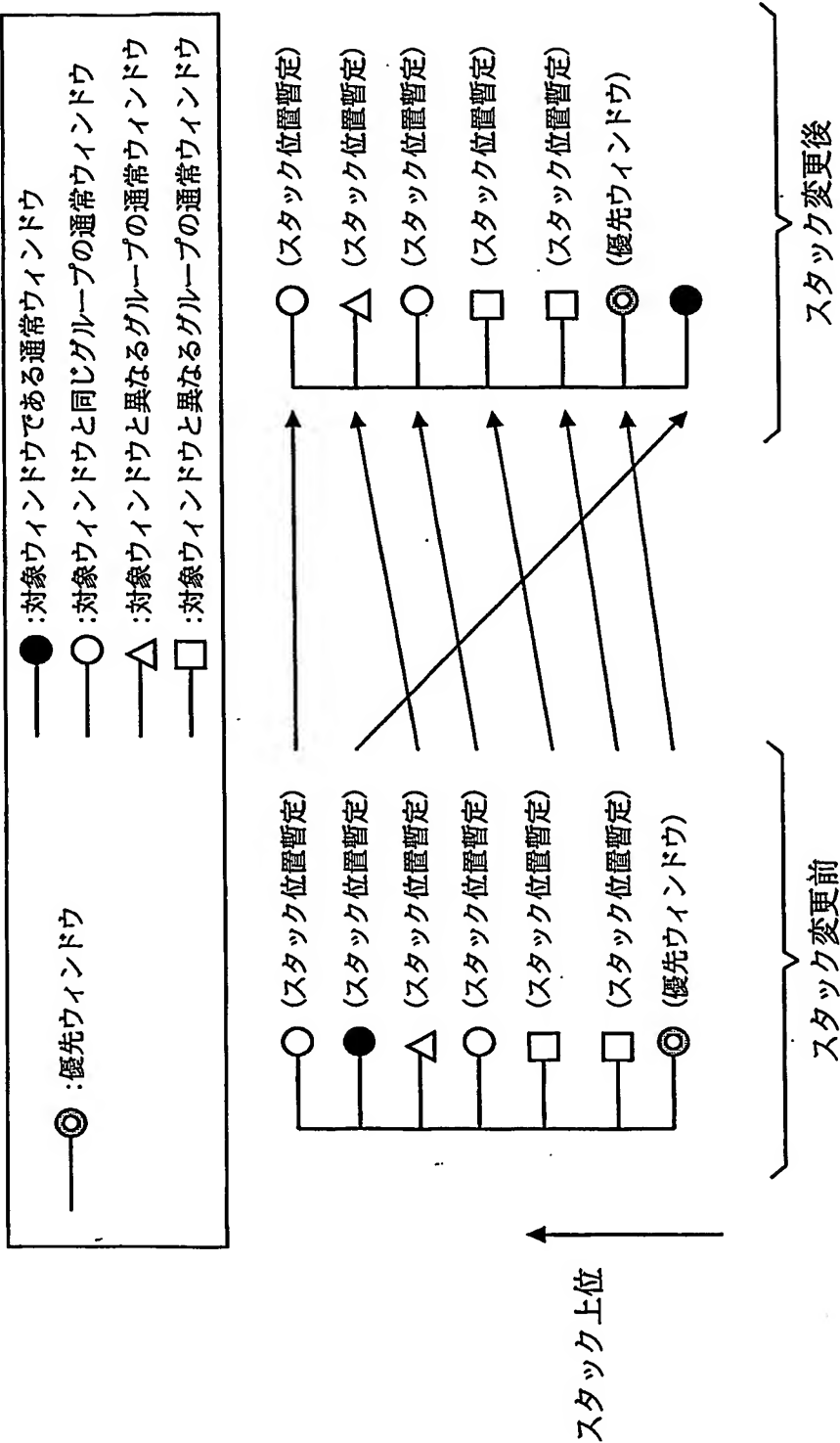


図7

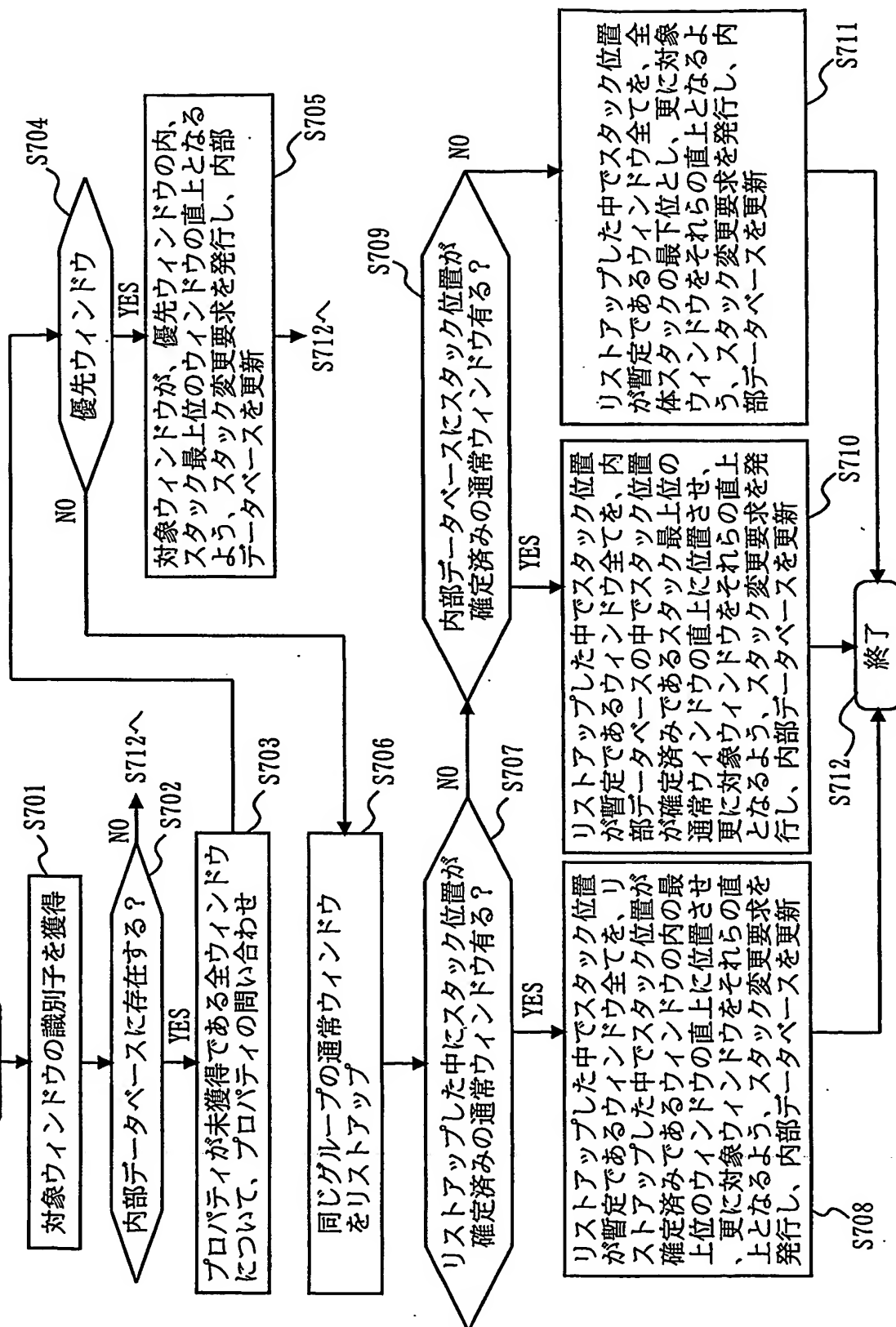
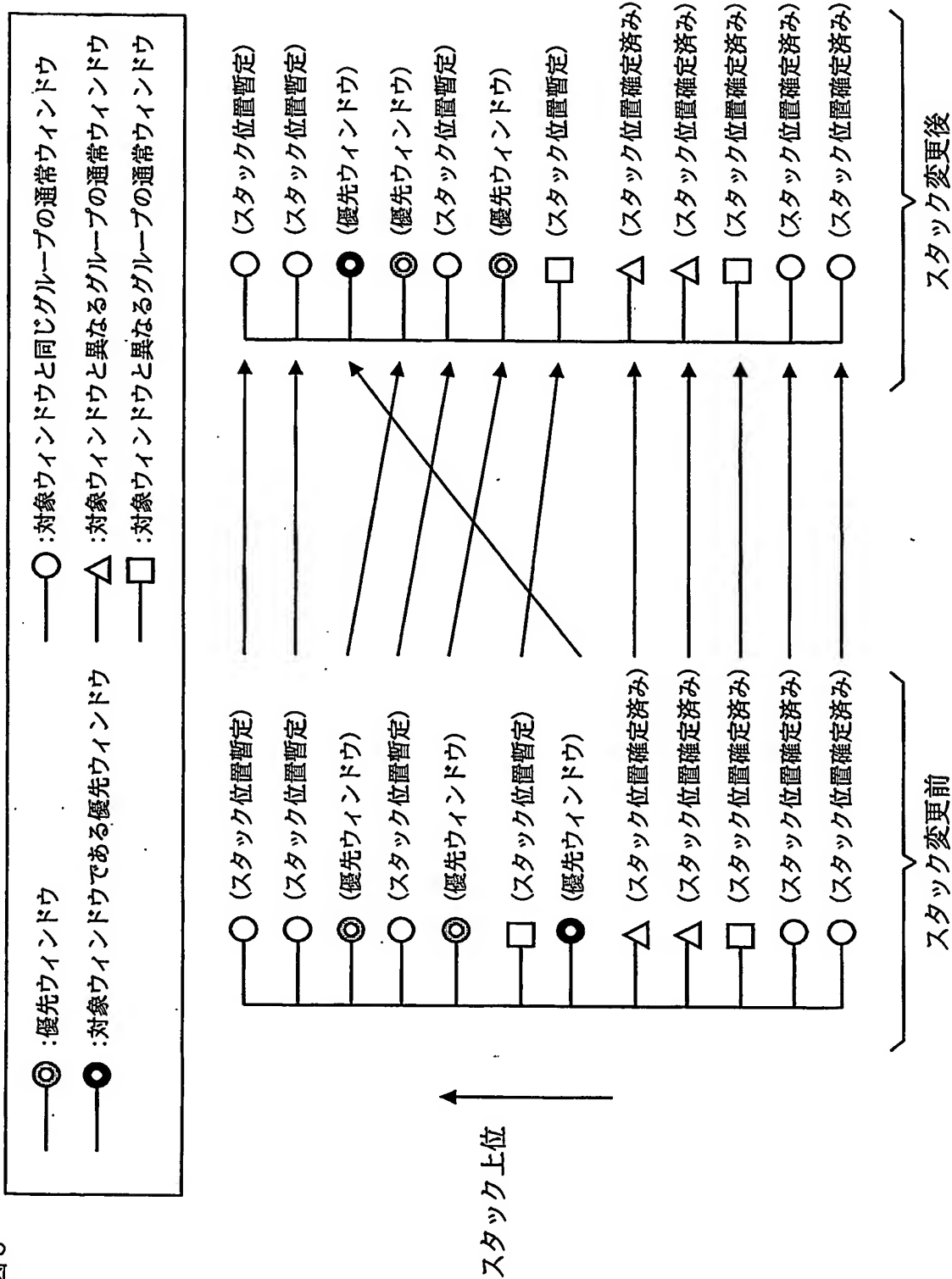


図 8



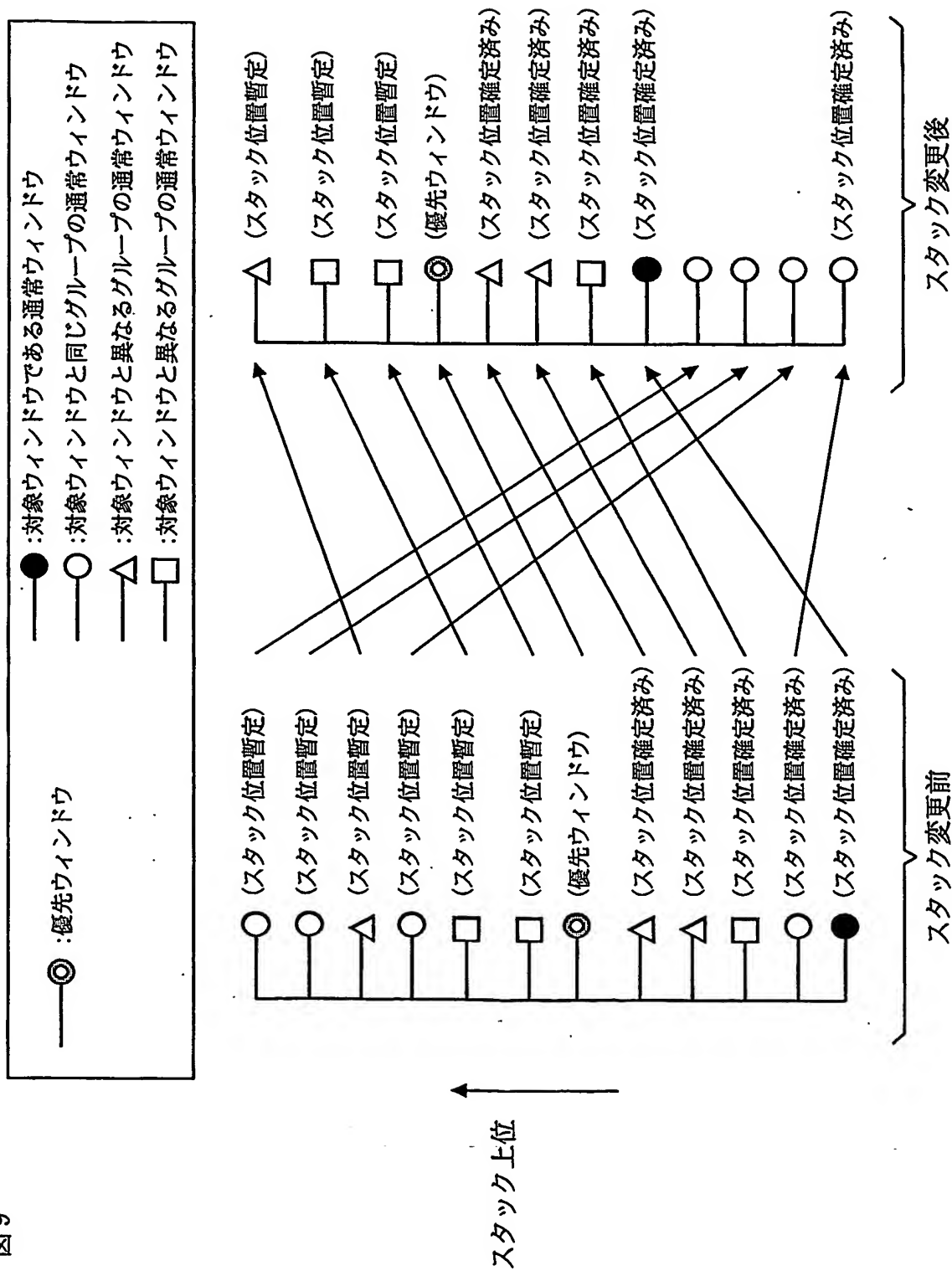
9  


図 10

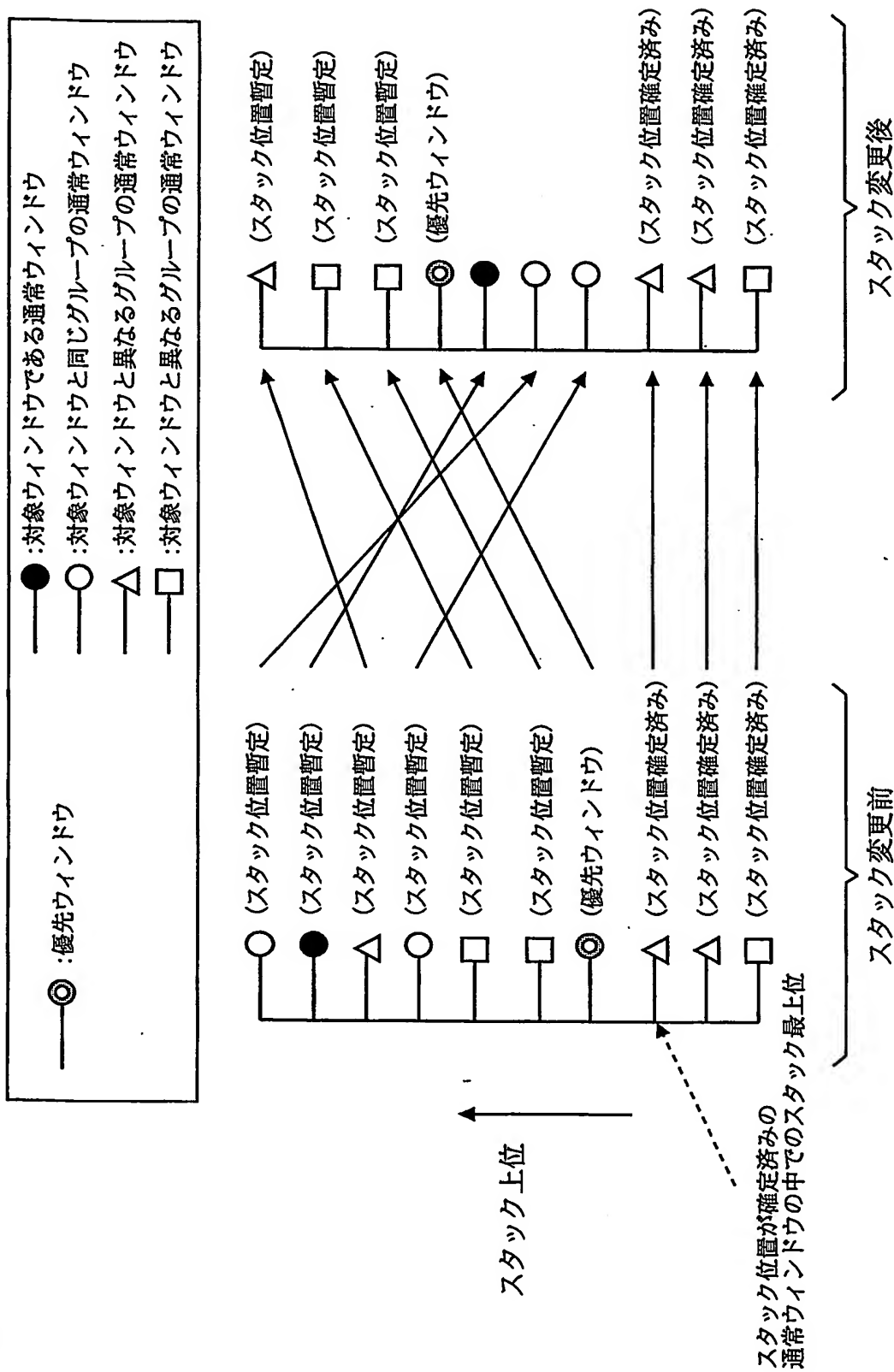


図 11

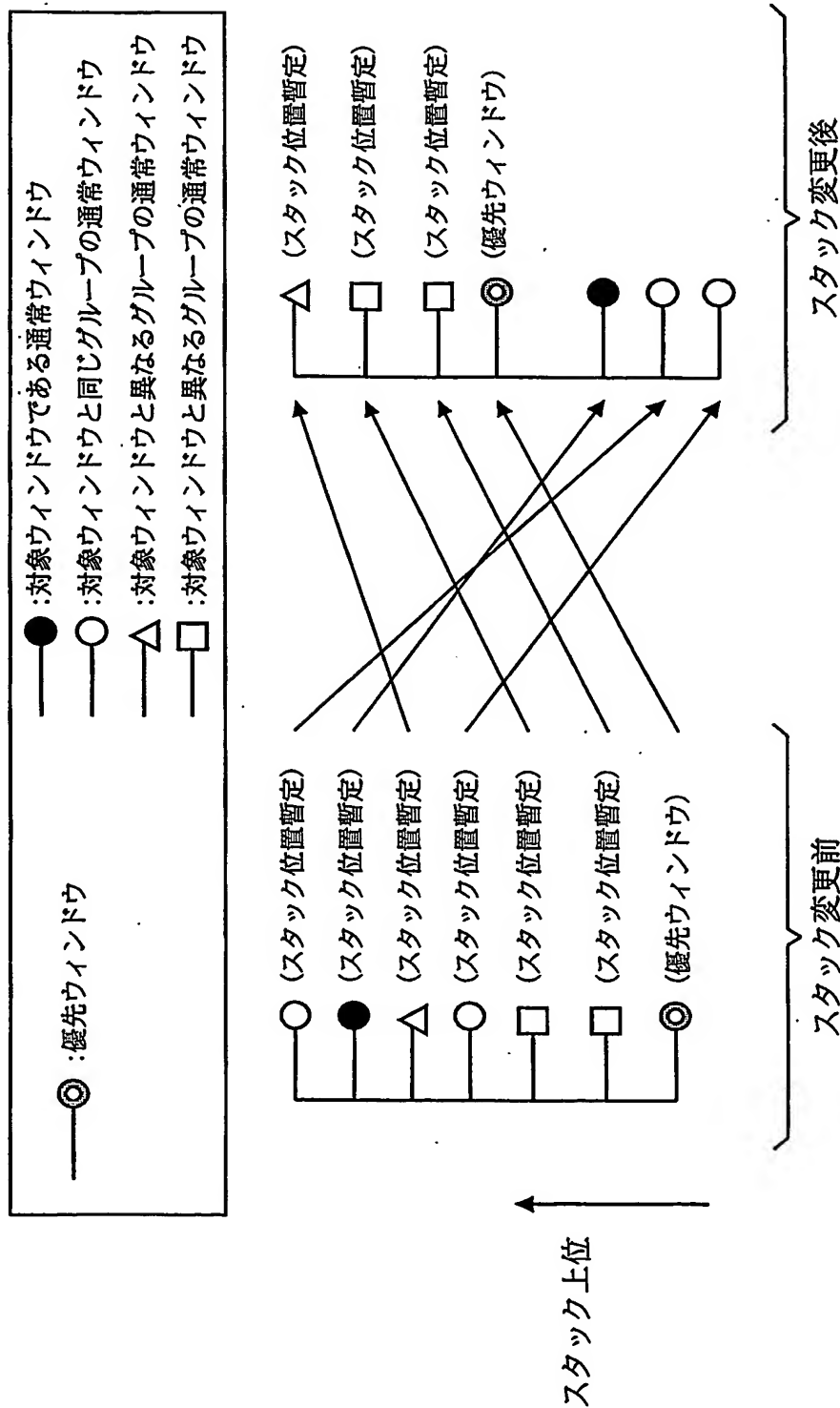


図 12

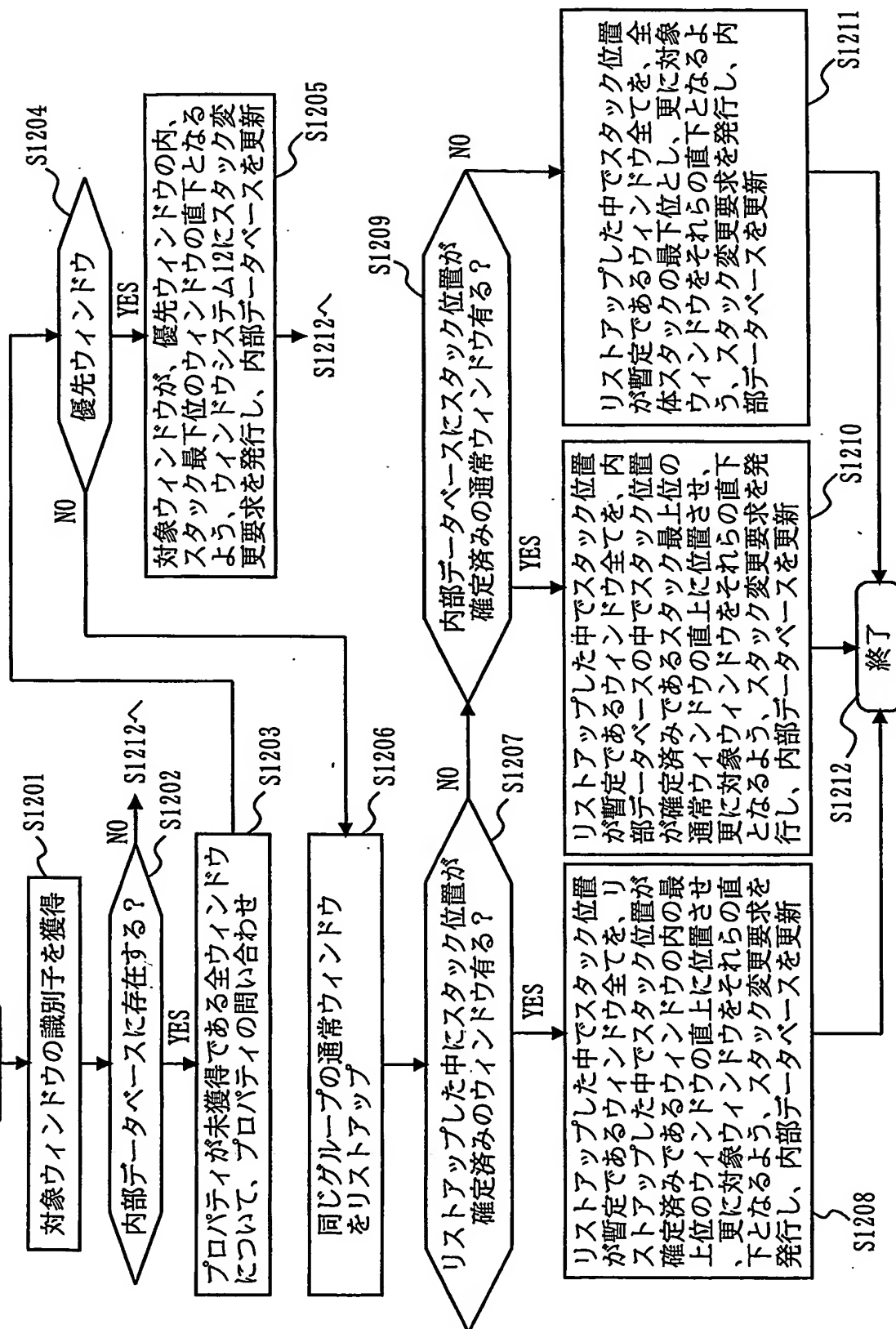




図 13

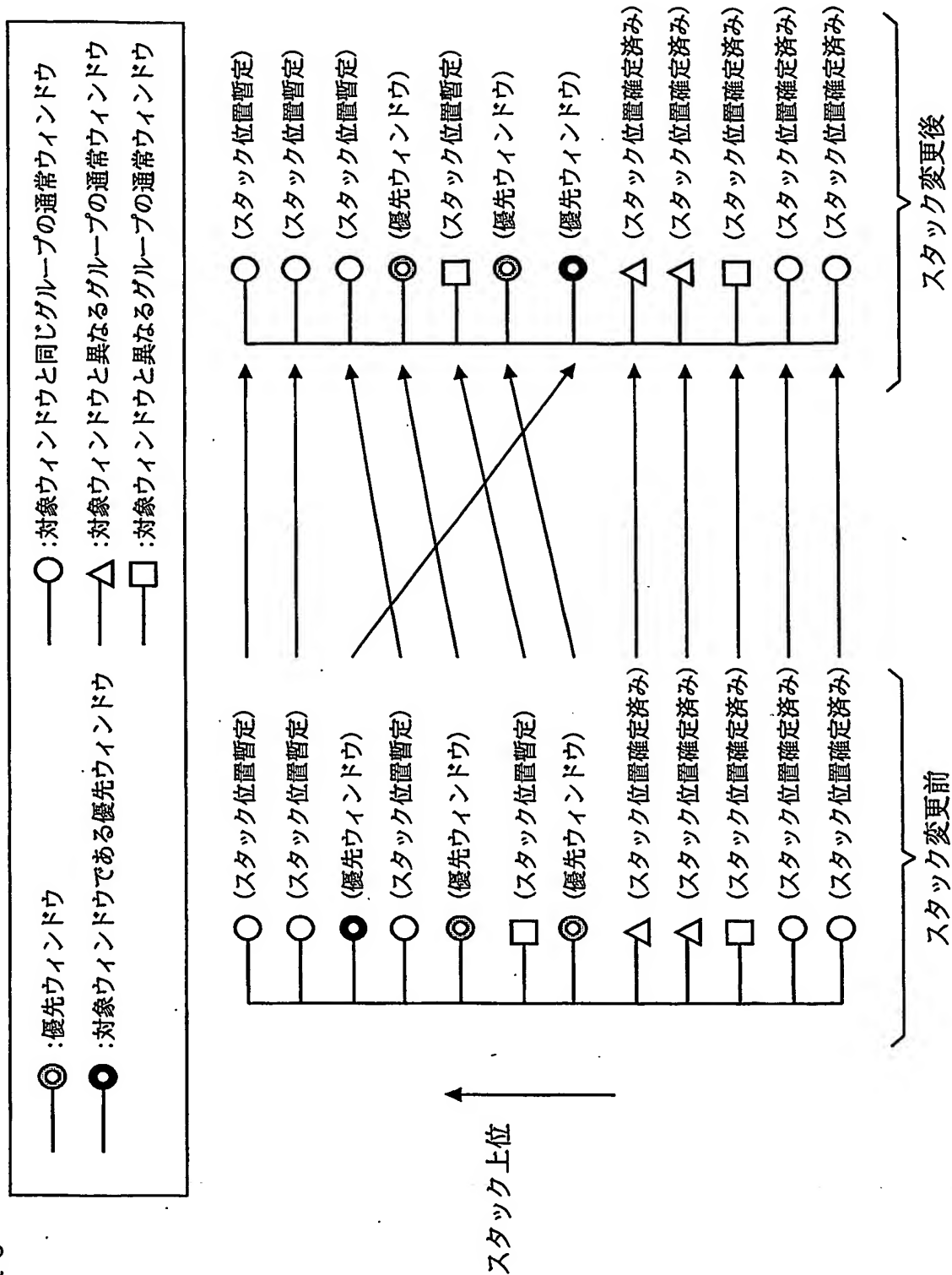


図 14

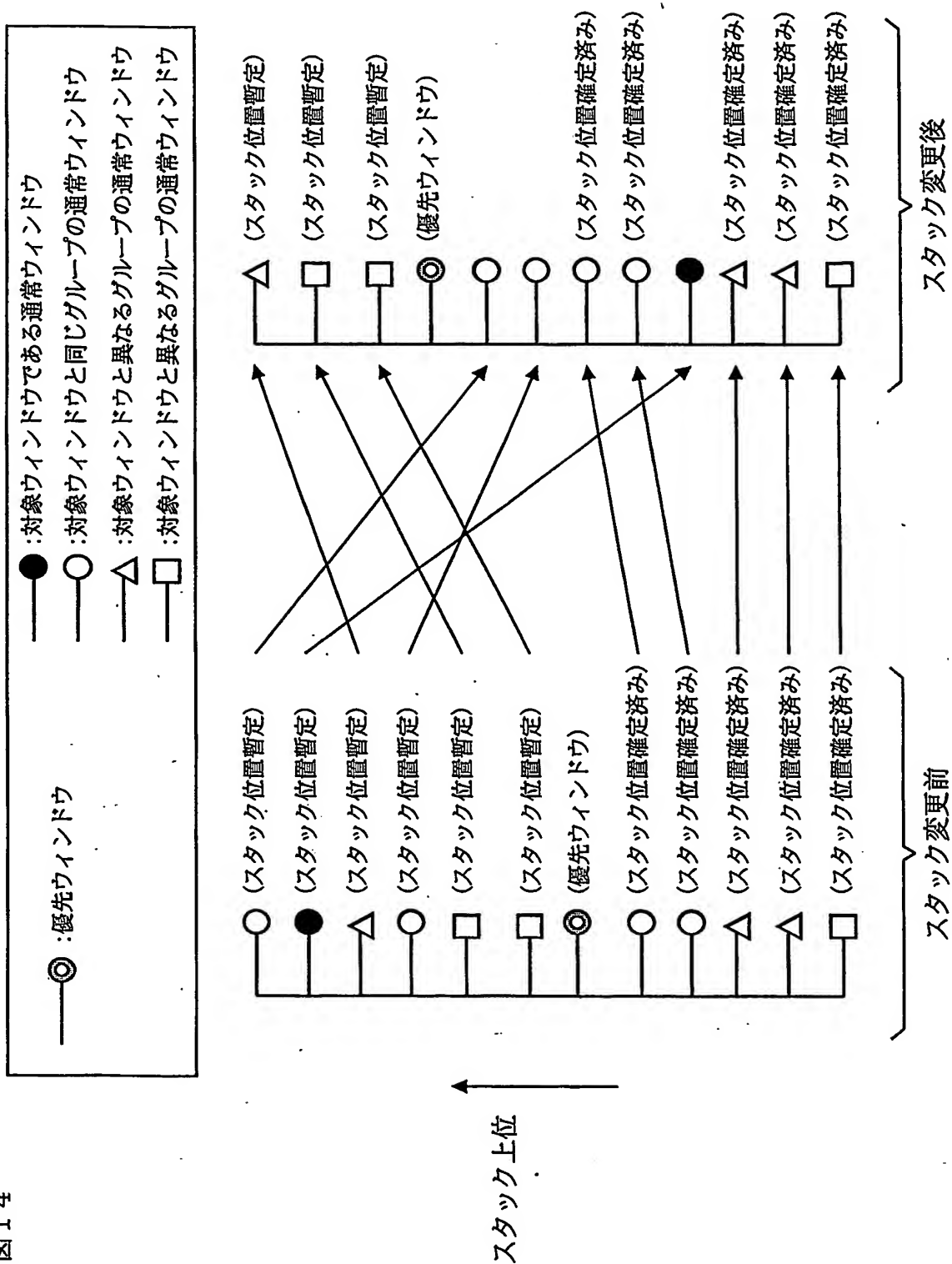


図 15

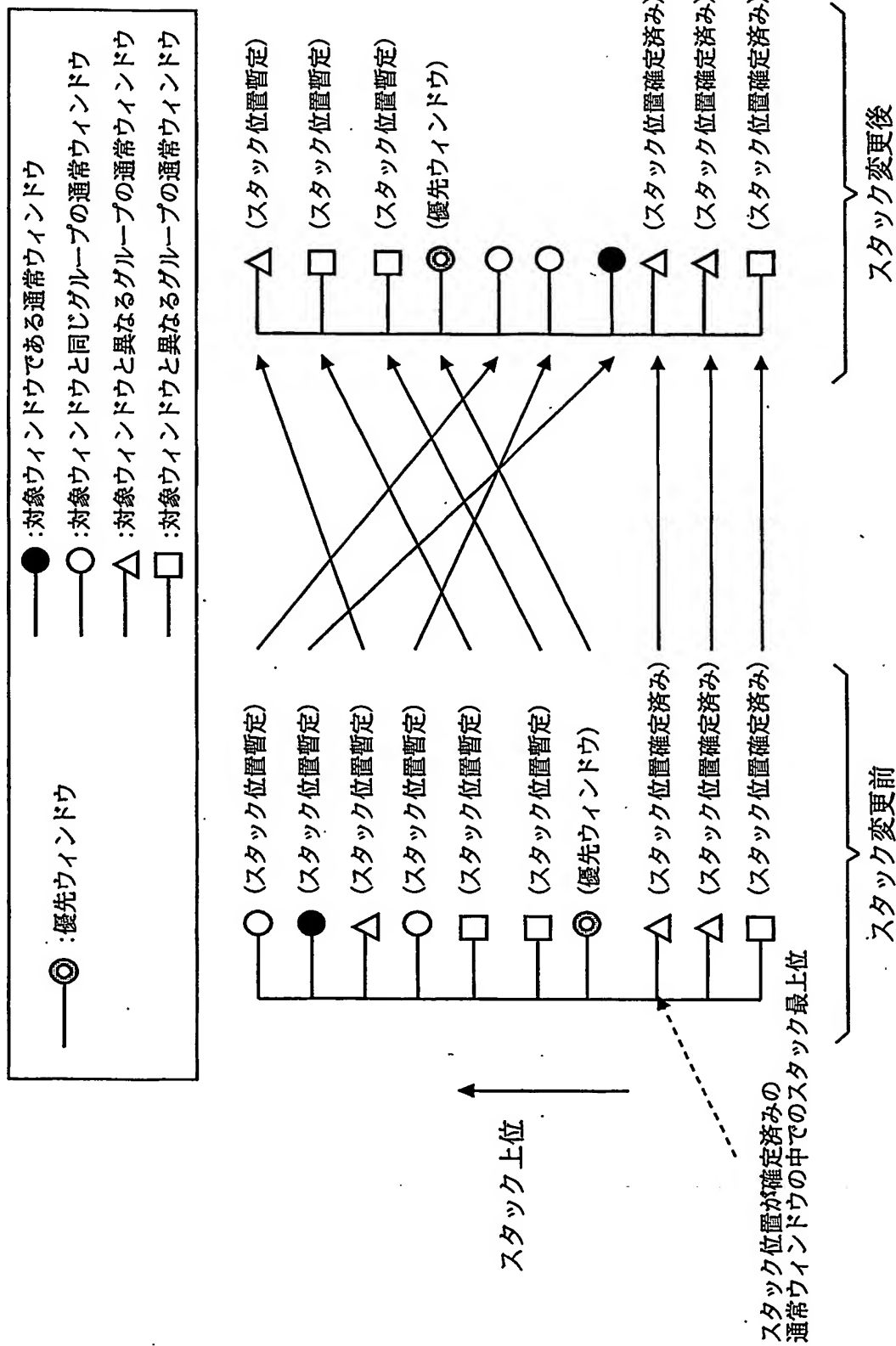


図 16

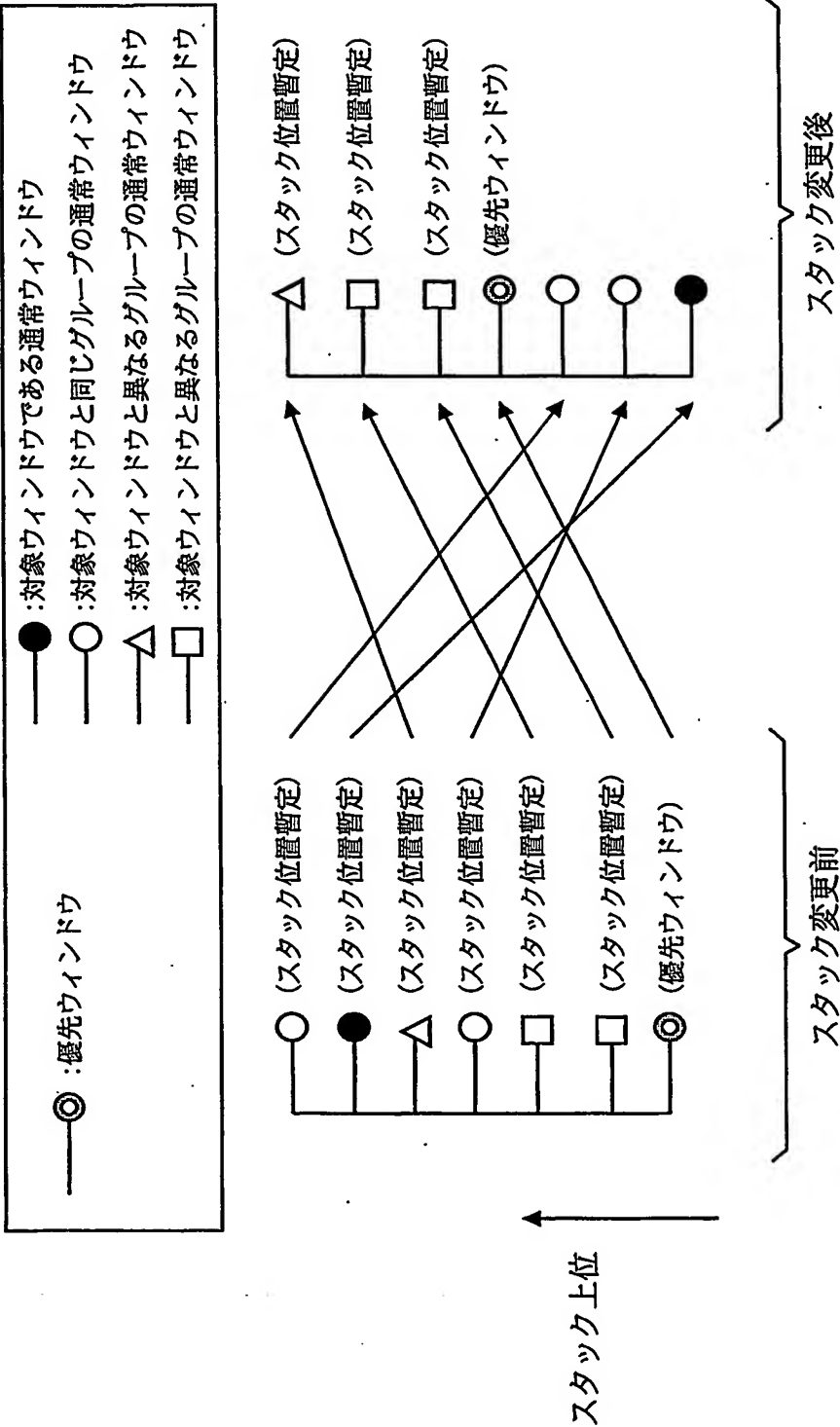


図 17

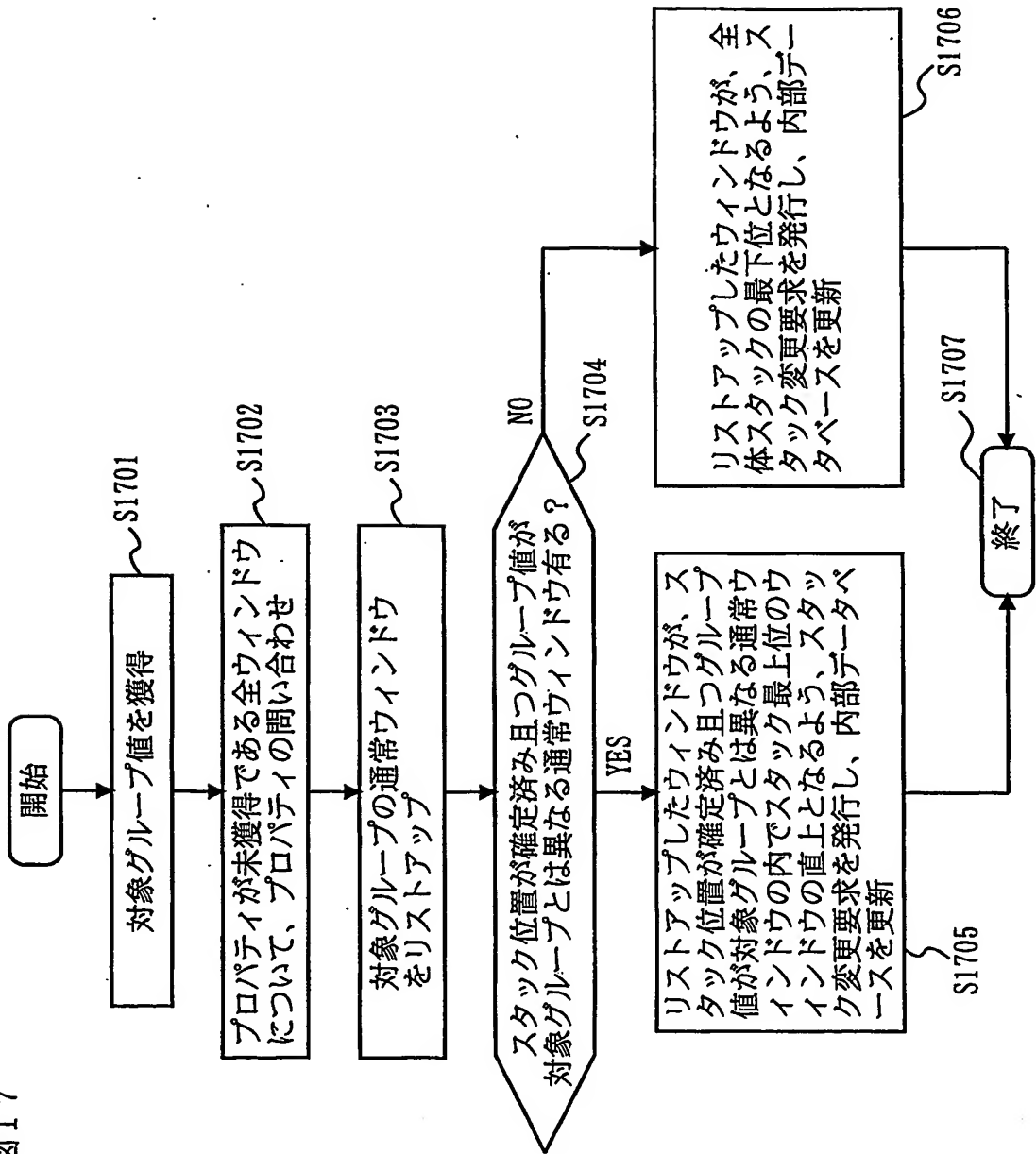




図19

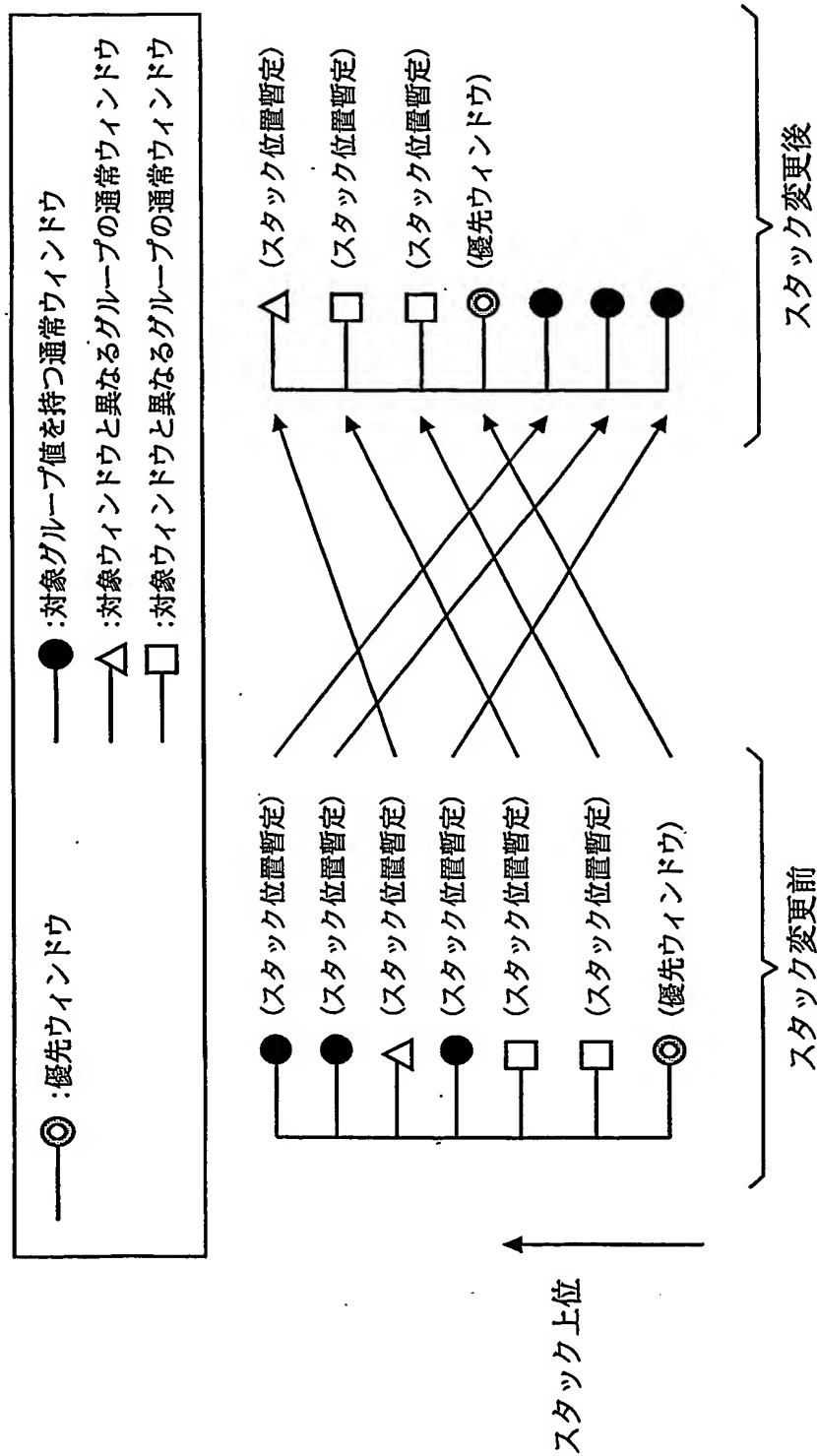


図 20

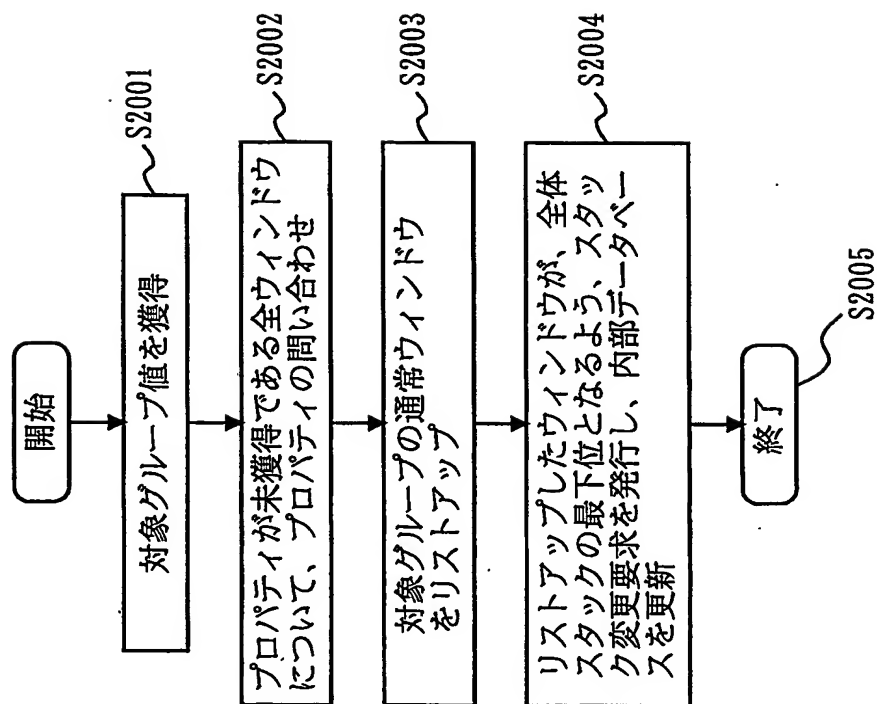






図 22

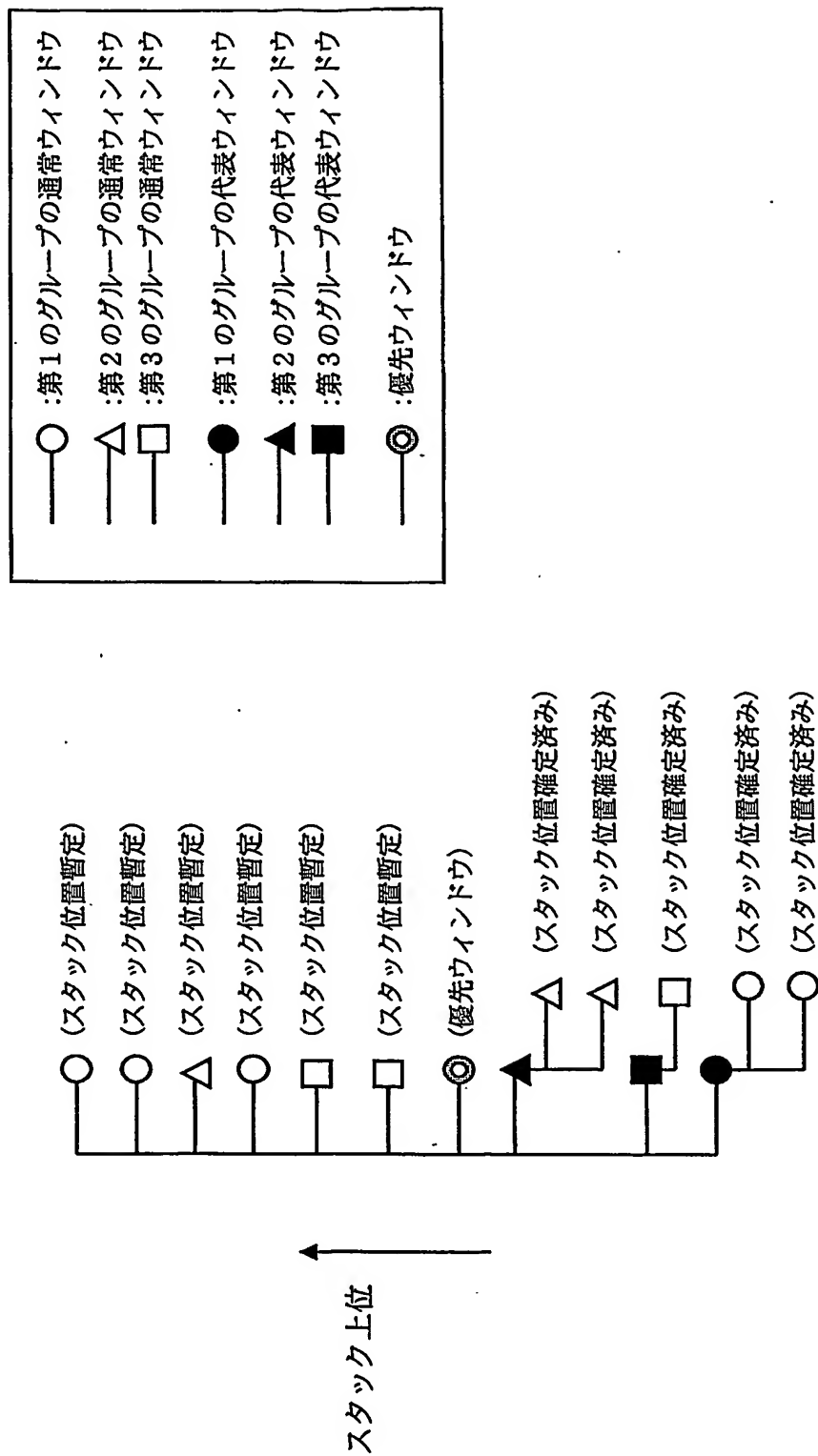


図 23

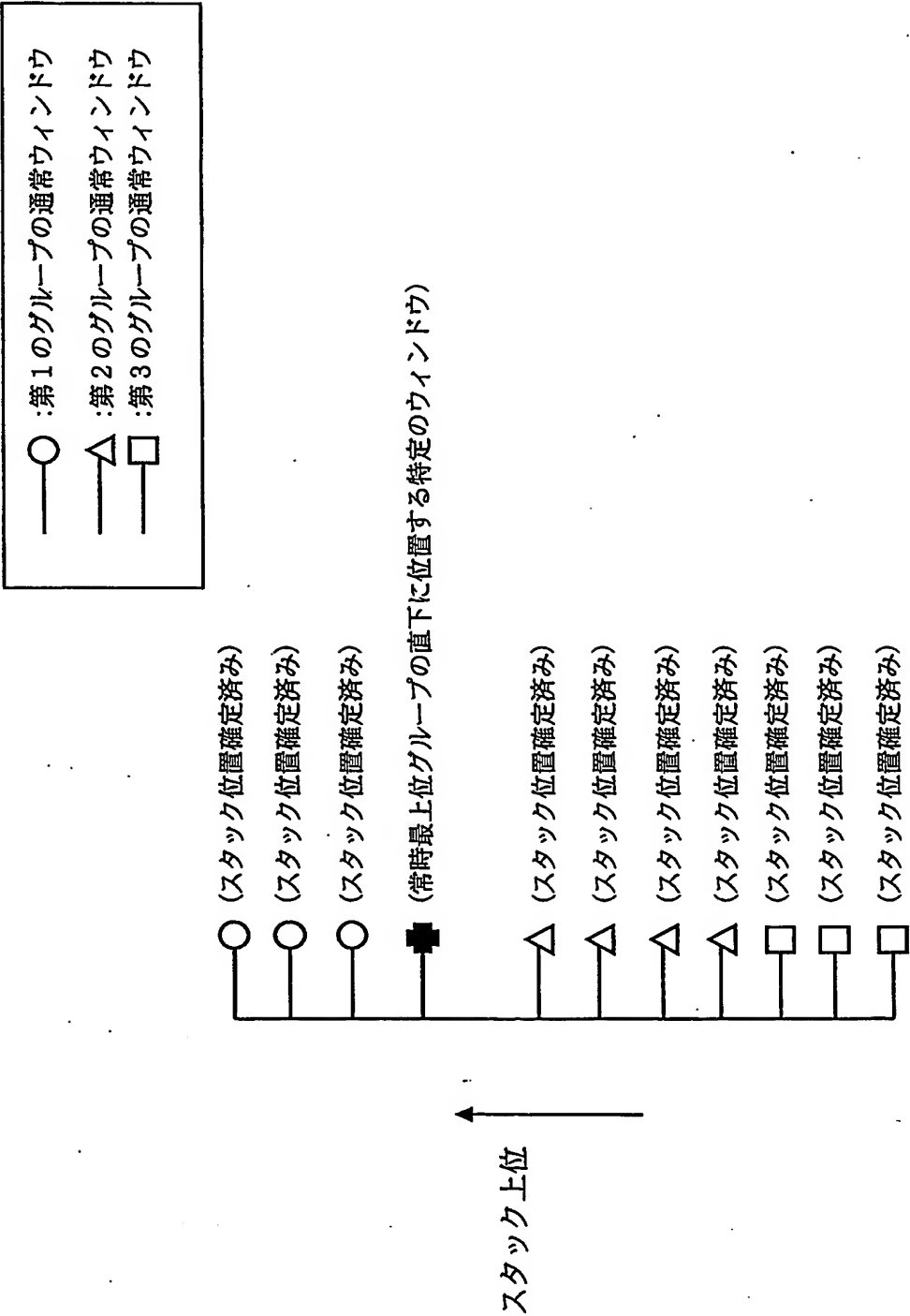


図 24

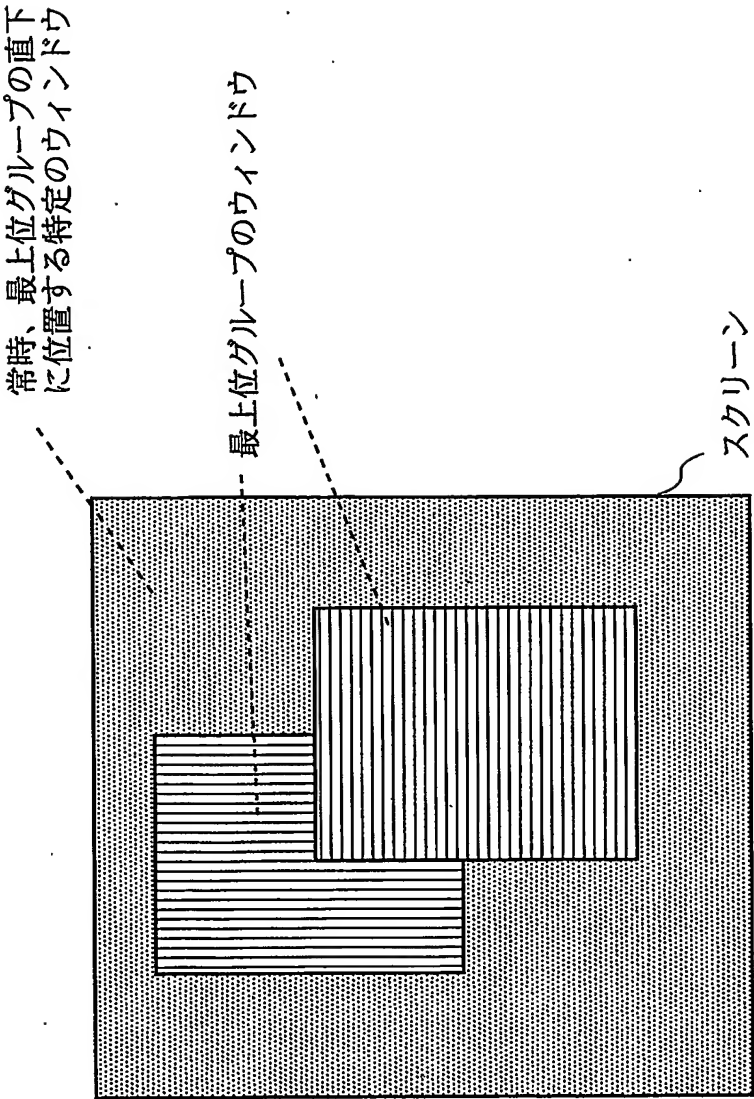
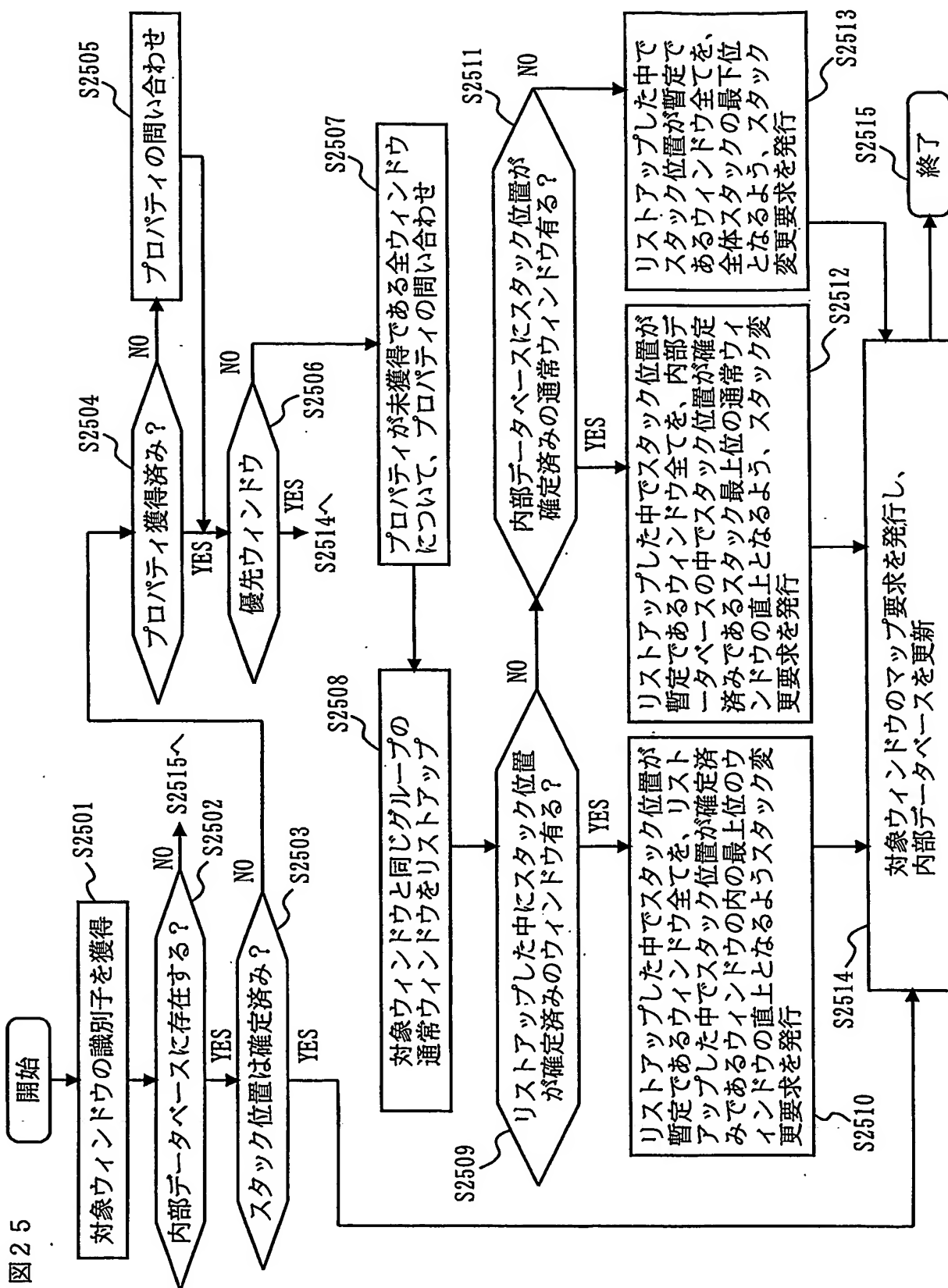


図 25



26

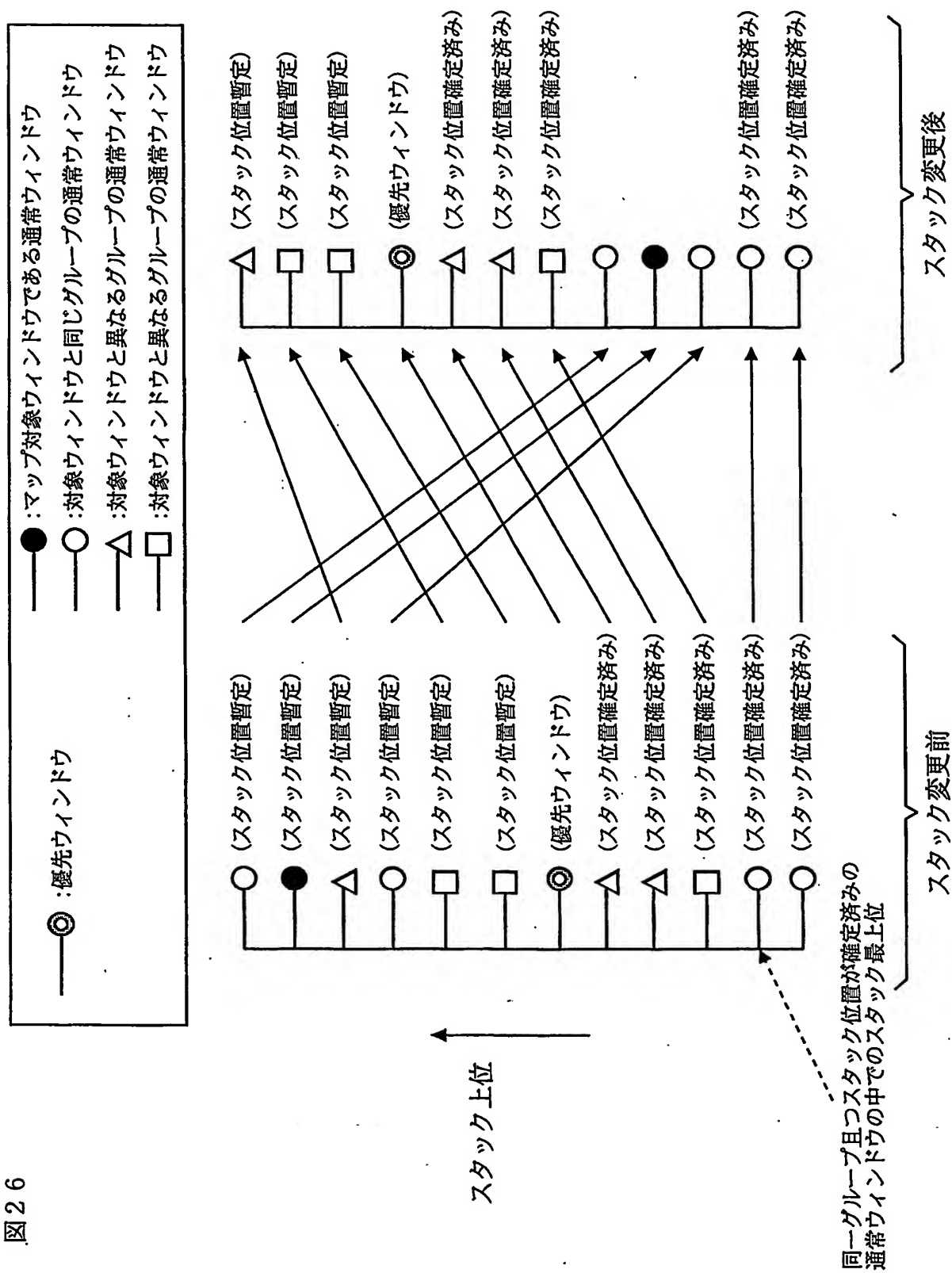
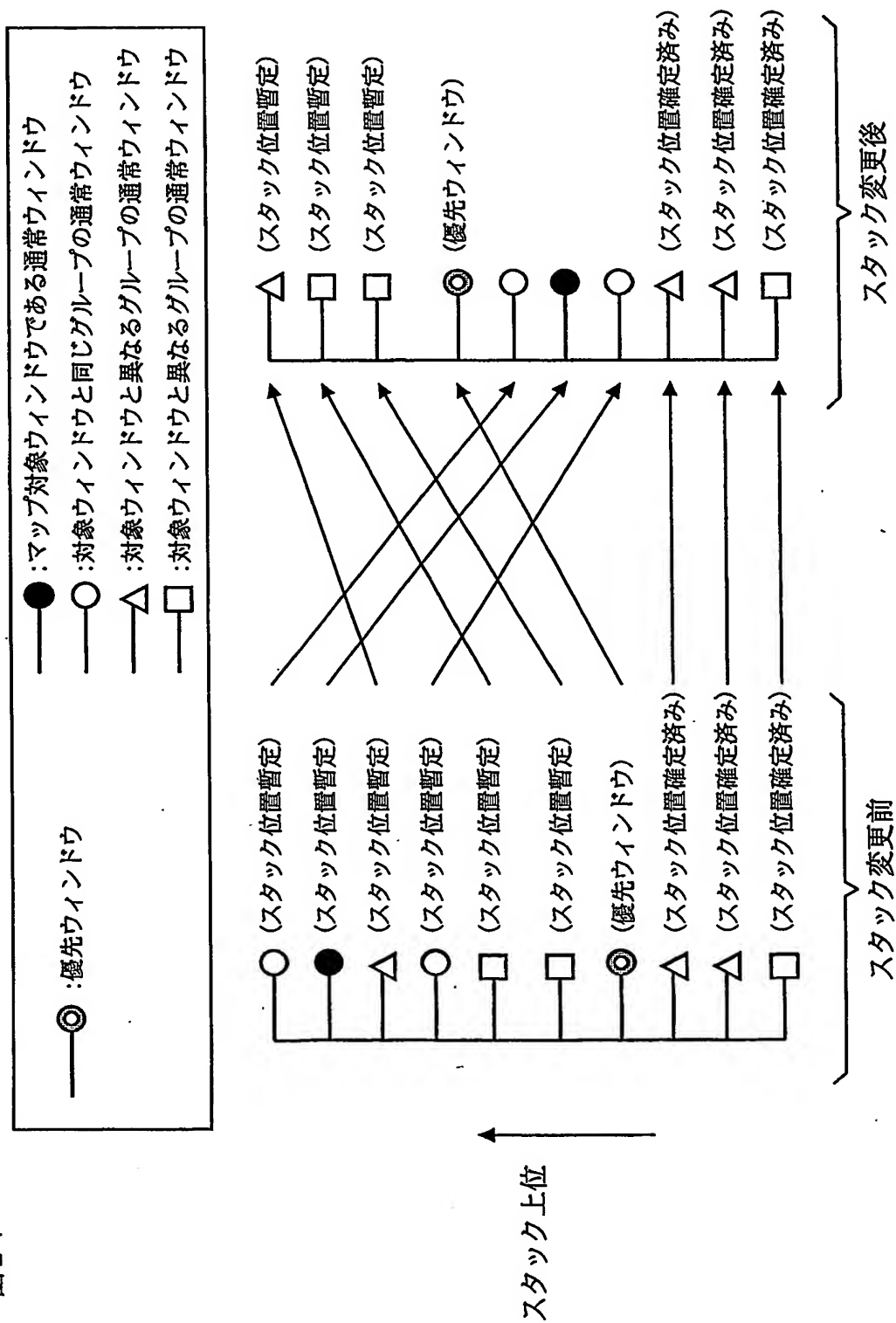


図 27



828

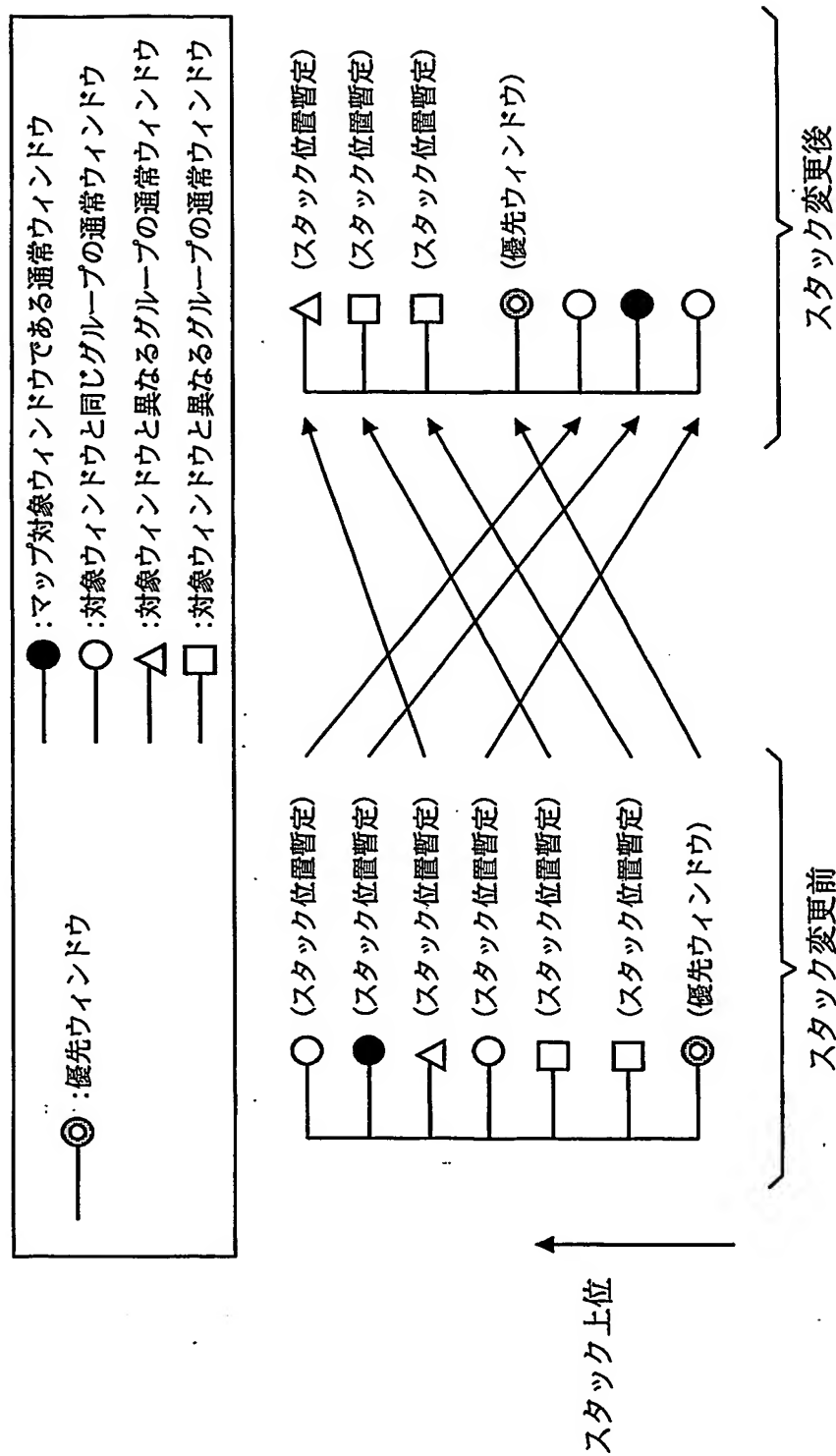




図29

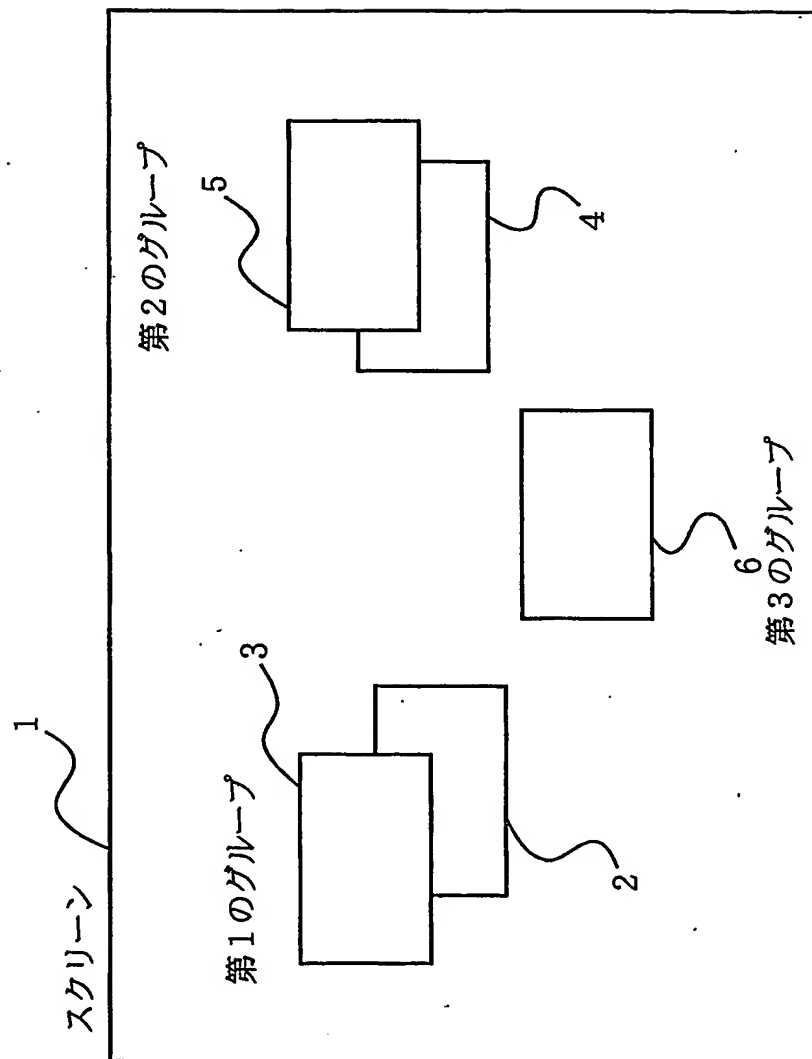


図30

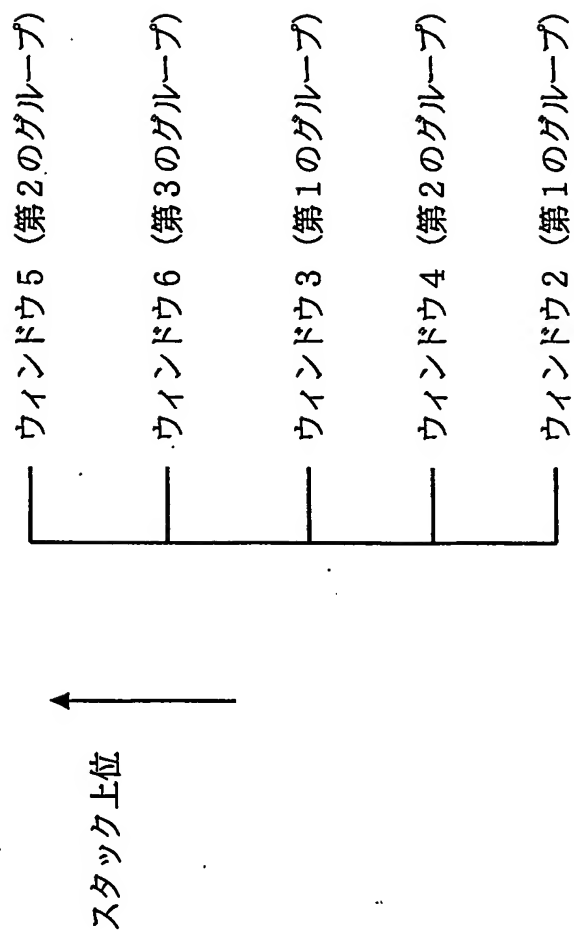
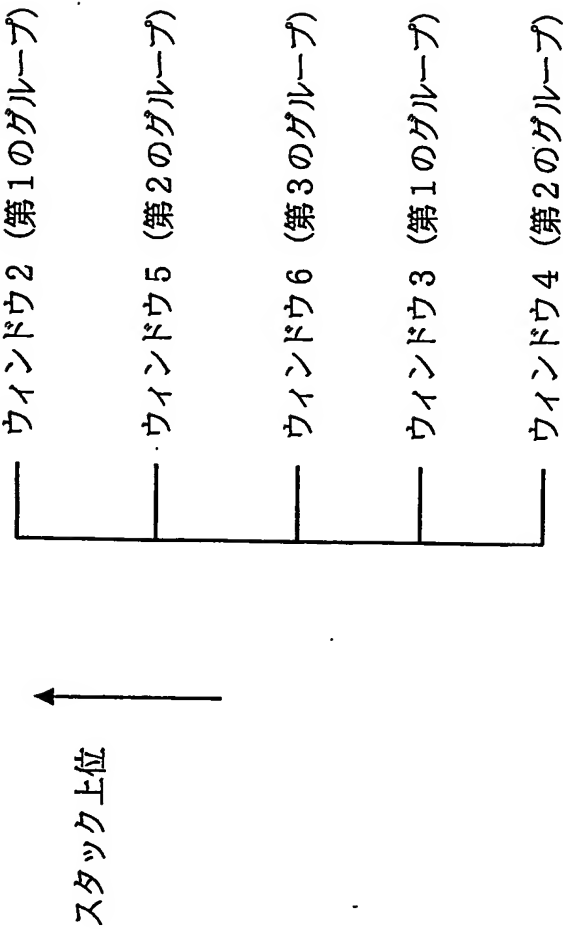


図 31



# INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2004/004414

## A. CLASSIFICATION OF SUBJECT MATTER

Int.Cl<sup>7</sup> G06F3/14, G09G5/14

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Int.Cl<sup>7</sup> G06F3/14, G06F3/00, G09G5/14

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1922-1996	Toroku Jitsuyo Shinan Koho	1994-2004
Kokai Jitsuyo Shinan Koho	1971-2004	Jitsuyo Shinan Toroku Koho	1996-2004

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	JP 02-244323 A (Fujitsu Ltd.), 28 September, 1990 (28.09.90), Full text; all drawings	1, 2, 4-8, 11-16
Y	Full text; all drawings (Family: none)	3, 9
X	JP 2001-060134 A (Hitachi, Ltd.), 06 March, 2001 (06.03.01), Full text; all drawings	1, 2, 4-8, 11-16
Y	Full text; all drawings (Family: none)	3, 9

☒ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

\* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

23 June, 2004 (23.06.04)

Date of mailing of the international search report

06 July, 2004 (06.07.04)

Name and mailing address of the ISA/  
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

## INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2004/004414

## C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	JP 05-282119 A (Nippon Telegraph And Telephone Corp.), 29 October, 1993 (29.10.93), Full text; all drawings	1, 2, 4-8, 10-16 3, 9
Y	Full text; all drawings (Family: none)	
X	JP 63-064121 A (Casio Computer Co., Ltd.), 22 March, 1988 (22.03.88), Full text; all drawings	1-8, 11-16 3, 9
Y	Full text; all drawings (Family: none)	
Y	JP 03-080323 A (Fujitsu Ltd.), 05 April, 1991 (05.04.91), Full text; all drawings (Family: none)	3
Y	JP 06-044031 A (NEC Corp.), 18 February, 1994 (18.02.94), Full text; all drawings (Family: none)	9

## A. 発明の属する分野の分類 (国際特許分類 (IPC))

Int. Cl<sup>7</sup> G06F 3/14, G09G 5/14

## B. 調査を行った分野

調査を行った最小限資料 (国際特許分類 (IPC))

Int. Cl<sup>7</sup> G06F 3/14, G06F 3/00, G09G 5/14

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報 1922-1996年  
 日本国公開実用新案公報 1971-2004年  
 日本国登録実用新案公報 1994-2004年  
 日本国実用新案登録公報 1996-2004年

国際調査で使用した電子データベース (データベースの名称、調査に使用した用語)

## C. 関連すると認められる文献

引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求の範囲の番号
X Y	JP 02-244323 A (富士通株式会社) 1990.09.28 全文、全図 全文、全図 (ファミリーなし)	1, 2, 4-8, 11-16 3, 9
X Y	JP 2001-060134 A (株式会社日立製作所) 2001.03.06 全文、全図 全文、全図 (ファミリーなし)	1, 2, 4-8, 11-16 3, 9

☒ C欄の続きにも文献が列挙されている。☐ パテントファミリーに関する別紙を参照。

## \* 引用文献のカテゴリー

「A」 特に関連のある文献ではなく、一般的技術水準を示すもの  
 「E」 国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの  
 「L」 優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献 (理由を付す)  
 「O」 口頭による開示、使用、展示等に関する文献  
 「P」 国際出願日前で、かつ優先権の主張の基礎となる出願

の日の後に公表された文献  
 「T」 国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの  
 「X」 特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの  
 「Y」 特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの  
 「&」 同一パテントファミリー文献

国際調査を完了した日

23.06.2004

国際調査報告の発送日

06.7.2004

国際調査機関の名称及びあて先

日本国特許庁 (ISA/JP)  
 郵便番号100-8915

東京都千代田区霞が関三丁目4番3号

特許庁審査官 (権限のある職員)

馬場 慎

5E

9743

電話番号 03-3581-1101 内線 3520

C (続き) . 関連すると認められる文献		
引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求の範囲の番号
X Y	J P 05-282119 A (日本電信電話株式会社) 1993. 10. 29 全文, 全図 全文, 全図 (ファミリーなし)	1, 2, 4-8, 10-16 3, 9
X Y	J P 63-064121 A (カシオ計算機株式会社) 1988. 03. 22 全文, 全図 全文, 全図 (ファミリーなし)	1-8, 11-16 3, 9
Y	J P 03-080323 A (富士通株式会社) 1991. 04. 05 全文, 全図 (ファミリーなし)	3
Y	J P 06-044031 A (日本電気株式会社) 1994. 02. 18 全文, 全図 (ファミリーなし)	9